Particle filters and smoothers in the dynamichazard package

Benjamin Christoffersen

August 20, 2019

This vignette covers the particle filters and smoothers implemented in the `dynamichazard` package in `R`. Some prior knowledge of particle filters is assumed. Doucet and Johansen (2009) provide a tutorial on particle filters and Kantas et al. (2015) cover parameter estimation with particle filters. See also Cappé et al. (2005) for a general introduction to Hidden Markov models. This vignette relies heavily on Fearnhead et al. (2010) and there is a big overlap between what is presented here and the paper.

1 Method

The models implemented in the package is survival analysis models for terminal events. These can be in discrete time where we have binary indicators $Y_{ik} = 1_{\{T_i \in (t_{k-1}, t_k]\}}$ which is one if the random event time of individual i denoted by $T_i \in (0, \infty)$ is in the interval $(t_{k-1}, t_k]$ and zero otherwise. It can also be in continuous time where we model the distribution of the event time of individual i, T_i, with a piecewise exponential distribution conditional on observable covariates and the path of a discrete latent variable. To be more concrete, the model is

$$\begin{array}{lll} y_{it} \sim g\left(y_{it} \vert \eta_{it}\right) & & \\ \boldsymbol{\eta}_t = \mathbf{X}_t \mathbf{R}^+ \boldsymbol{\alpha}_t + \boldsymbol{o}_t + \mathbf{Z}_t \boldsymbol{\omega} & & i = 1, \dots, n_t \\ \boldsymbol{\alpha}_t = \mathbf{F} \boldsymbol{\alpha}_{t-1} + \mathbf{R} \boldsymbol{\epsilon}_t & \boldsymbol{\epsilon}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\right), & t = 1, \dots, d \\ & \boldsymbol{\alpha}_0 \sim \mathcal{N}\left(\boldsymbol{\mu}_0, \mathbf{Q}_0\right) & \end{array} \tag{1}$$

where I denote the conditional densities as $g_t\left(\boldsymbol{y}_t \vert \boldsymbol{\alpha}_t\right) = g\left(\boldsymbol{y}_t \vert \mathbf{X}_t \mathbf{R}^+ \boldsymbol{\alpha}_t + \boldsymbol{o}_t + \mathbf{Z}_t \boldsymbol{\omega}\right)$ and $f\left(\boldsymbol{\alpha}_t \vert \boldsymbol{\alpha}_{t-1}\right)$. For each $t = 1, \dots, d$, we a have risk set given by $R_t \subseteq \{1, 2, \dots, n\}$. Further, we let $n_t = \vert R_t \vert$ denote the number of observation at risk at time t and $n_{\max} = \max_{t \in \{1, \dots d\}} n_t$. The observed outcomes are denoted by $\boldsymbol{Y}_t = \{y_{it}\}_{i \in R_t}$. $\mathbf{X}_t$ is the design matrix of the covariates and $\boldsymbol{\alpha}_t$ is the state vector containing the time-varying coefficients. The $\mathbf{Z}_t$ is the design matrix for the covariates with time-invariant coefficients and $\boldsymbol{\omega}$ are the corresponding coefficients.

The i'th row of $\mathbf{X}_t$ is $\boldsymbol{x}_{it}$, $\boldsymbol{x}_{it}, \boldsymbol{\epsilon}_t \in \mathbb{R}^r$, $\boldsymbol{\mu}_0, \boldsymbol{\alpha}_t \in \mathbb{R}^p$, $\mathbf{F} \in \mathbb{R}^{p \times p}$, $\mathbf{Q} \in \mathbb{R}^{r \times r}$ is a positive definite matrix, $\mathbf{Q}_0 \in \mathbb{R}^{p \times p}$ is a positive definite matrix, $\boldsymbol{o}_t$s are known offsets, and $\mathbf{R}$ is a $p \times r$ matrix with $p \geq r$ which contains a subset of r columns of $\mathbf{I}_p$. We will order the entires of $\mathbf{R}$ such that the first r columns are the first r columns of $\mathbf{I}_p$. I.e.,

$$\mathbf{R} = \begin{pmatrix} \mathbf{I}_r \\ \mathbf{0} \end{pmatrix}$$

Superscript + denotes the Moore-Penrose inverse, $\mathbf{R}^+ = \mathbf{R}^\top$ and $\mathbf{R}$ is left inverse (i.e., $\mathbf{R}^\top\mathbf{R} = \mathbf{I}_r$). $\mathbf{R}\mathbf{R}^\top$ is a $p \times p$ diagonal matrix where the r first diagonal entries has value 1 and the rest of the diagonal entries are zero. The data sets we are working with have $n_{\max} \gg p \geq r$ (e.g. $n_{\max} = 1000$ and $r = 5$). We let

$$\boldsymbol{\xi}_t = \mathbf{R}^+\boldsymbol{\alpha}_t$$

The above allows us to have an oth order vector autoregressions, VAR(o), by settings

$$\begin{aligned}
\boldsymbol{\alpha}_t &= (\boldsymbol{\xi}_t, \boldsymbol{\xi}_{t-1}, \dots, \boldsymbol{\xi}_{t-o+1}) \\
\mathbf{F} &= \begin{pmatrix} \mathbf{F}_1 & \cdots & \cdots & \mathbf{F}_{o-1} & \mathbf{F}_o \\ \mathbf{I}_r & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_r & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I}_r & \mathbf{0} \end{pmatrix}, \qquad \mathbf{F}_i \in \mathbb{R}^{r\times r}
\end{aligned}$$

I will use a particle filter to get a discrete approximation of the conditional distribution of $\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_d$ given the outcomes $\boldsymbol{y}_{1:d} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_d\}$ and use an EM-algorithm to estimate $\mathbf{Q}$, $\boldsymbol{\omega}$, and $\boldsymbol{\mu}_0$. One choice of smoother is shown in Fearnhead et al. (2010) and another is the generalized two-filter smoother shown by Briers et al. (2009). The rest of vignette is structured as follows: first I give a brief introduction to the implemented particle filters and smoothers. Then I cover what the effect is of some the arguments to particle functions in `R` in the packages. The implemented particle filter and smoother from Fearnhead et al. (2010) is presented next, followed by the used EM-algorithm and the smoother suggested by Briers et al. (2009). The last section covers the implemented approximations of the gradient and observed information matrix.

1.1 Overview

As a gentle introduction before the next sections, we will start by recalling an application of importance sampling, use this to motivate particle filtering, and give a brief idea of the implemented particle smoothers. Suppose we want to approximate a density $c(x) = \zeta\tilde{c}(x)$ where we only know $\tilde{c}(x)$ and not the normalization constant ζ. One way to approximate this density is to

- sample $x_1, x_2, \dots, x_N$ from a distribution with density $b(x)$.
- Compute the unnormalized weights $\bar{w}_i = \tilde{c}(x_i)/b(x_i)$.
- Normalize the weights $w_i = \bar{w}_i / \sum_{i=1}^N \bar{w}_i$.

This gives us the following discrete approximation of the density

$$c(x) \approx \sum_{i=1}^N w_i \delta_{x_i}(x)$$

where δ_x is the Dirac delta function which has unit point mass at x. This is directly applicable to the model in Equation (1) as at time 1 we want to approximate

$$p(\boldsymbol{\alpha}_1|\boldsymbol{y}_1) = \frac{g_1(\boldsymbol{y}_1|\boldsymbol{\alpha}_1)\int f(\boldsymbol{\alpha}_1|\boldsymbol{a}_0)\,\phi(\boldsymbol{a}_0|\boldsymbol{\mu}_0, \mathbf{Q}_0)\,\mathrm{d}\boldsymbol{a}_0}{p(\boldsymbol{y}_1)}$$

where $\phi(\cdot|\boldsymbol{m},\mathbf{M})$ is the density function of a multivariate normal distribution with mean $\boldsymbol{m}$ and covariance matrix $\mathbf{M}$. We can easily evaluate the numerator for each $\boldsymbol{\alpha}_1$ but not the normalization constant, $p(\boldsymbol{y}_1)$.

The extension to a particle filter (which I will call a forward particle filter) is that at time 2 we want to approximate

$$p(\boldsymbol{\alpha}_{1:2}|\boldsymbol{y}_{1:2}) = p(\boldsymbol{\alpha}_1|\boldsymbol{y}_1)\frac{g_2(\boldsymbol{y}_2|\boldsymbol{\alpha}_2) f(\boldsymbol{\alpha}_2|\boldsymbol{\alpha}_1)}{p(\boldsymbol{y}_2|\boldsymbol{y}_1)}$$

Now, we can use the discrete approximation at time 1 of $p(\boldsymbol{\alpha}_1|\boldsymbol{y}_1)$, sample $\boldsymbol{\alpha}_2$ given each sampled $\boldsymbol{\alpha}_1$, and apply importance sampling again. We can repeat this with similar arguments at time $3, 4, \dots, d$ giving us an approximation of $p(\boldsymbol{\alpha}_{1:d}|\boldsymbol{y}_{1:d})$. We will call the last element of a sampled path at time t a particle. Further, we will denote the jth particle at time t and its associated weight by $\boldsymbol{\alpha}_t^{(j)}$ and $w_t^{(j)}$ respectively.

One issue that may arise is that our samples (particles) may degenerate so essentially only one sampled path of $\boldsymbol{\alpha}_{1:d}$ has any weight in the end. To avoid this, we may introduce a re-sampling step. One way to re-sample is using the weights and letting the re-sampling weights be $\beta_{t+1}^{(j)} = w_t^{(j)}$ where $\beta_{t+1}^{(j)}$ is the re-sampling weight of particle j at time t. We then sample with replacement using $\beta_{t+1}^{(j)}$. Another option when we re-sample the particles from time t is to use the information of the outcomes at time $t+1$, $\boldsymbol{y}_{t+1}$. This is called an auxiliary particle filter and is introduced by Pitt and Shephard (1999).

However, we may end up with few or only one unique value at the early time points (say $\boldsymbol{\alpha}_1$) when we re-sample. Thus, it will be useful to use a smoother to get a better approximation of the marginal density $p(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:d})$. To do so, one idea is to use the two-filter formula from Kitagawa (1994). Though, this requires that we can evaluate $p(\boldsymbol{y}_{t:d}|\boldsymbol{\alpha}_t)$. It turns out that we can approximate this up to a constant which is just what need. This is covered in further details in Section 6.

The approximation uses a particle filter which is run backwards in time and which approximates an artificial distribution. The arguments for the backward particle filter is very similar to the forward particle filter presented above. The kth particle in the backward particle filter at time t, its re-sampling weight, and the associated weight will be denoted by respectively $\widetilde{\boldsymbol{\alpha}}_t^{(k)}$, $\widetilde{\beta}_{t-1}^{(k)}$ and $\widetilde{w}_t^{(k)}$. The final ith smoothed particle and weight at time t will be denoted by $\widehat{\boldsymbol{\alpha}}_t^{(i)}$ and $\widehat{w}_t^{(i)}$. The latter gives us the following approximation of the marginal density of $\boldsymbol{\alpha}_t \mid \boldsymbol{y}_{1:d}$

$$p(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:d}) \approx \sum_{i=1}^{N_S} \widehat{w}_t^{(i)} \delta_{\widehat{\boldsymbol{\alpha}}_t^{(i)}}(\boldsymbol{\alpha}_t)$$

if we sampled N_S smoothed particles at time t. The smoothing algorithm from Fearnhead et al. (2010) is shown in Algorithm 1, the forward particle filter is shown Algorithm 2, and the backward particle filter is shown in Algorithm 3.

1.2 Methods in the Package

The `PF_EM` method in the `dynamichazard` package contains an implementation of the described methods. You specify the number of particles by the `N_first`, `N_fw_n_bw` and `N_smooth` argument for respectively the N_f, N and N_s in the Algorithm 1-3. We may want more particles in the smoothing step, $N_s > N$, as pointed out in the discussion in Fearnhead et al. (2010, p. 460-461). Further, selecting $N_f > N$ may be preferable to ensure coverage of the state space at time 0 and $d + 1$.

We do not need to sample the time 0 and $d + 1$ particles. Instead we can make a special proposal distribution for time 1 and time d. This is not implemented though...

The `method` argument specify how the filters are set up. The argument can take the following values

- `"bootstrap_filter"` for a bootstrap filter. This is where we sample using Equation (5), (11) and (14). This is fast but the proposal distribution may be a poor approximation of the distribution we want to target.
- `"PF_normal_approx_w_cloud_mean"` and `"AUX_normal_approx_w_cloud_mean"` for the Taylor approximation of the conditional density of $\boldsymbol{y}_t$ made using the mean of the parent particles and/or mean of the child particles. See Section 2. The `PF` and `AUX` prefix specifies whether or not the auxiliary version should be used.
- `"PF_normal_approx_w_particles"` and `"AUX_normal_approx_w_particles"` for the Taylor approximation of the conditional density of $\boldsymbol{y}_t$ made using the parent and/or child particle. See Section 2. The `PF` and `AUX` prefix specifies whether or not the auxiliary version should be used.

The smoother is selected with the `smoother` argument. `"Fearnhead_O_N"` gives the smoother in Algorithm 1 and `"Brier_O_N_square"` gives the smoother in Algorithm 4. The *Systematic Re-sampling* (Kitagawa, 1996) is used in all re-sampling steps. See Douc and Cappé (2005) for a comparison of re-sampling methods. The rest of the arguments to `PF_EM` are similar to those of the `ddhazard` function.

It is not clear what will give the best performance for a given data set at a fixed computation cost. An advice is to use the `trace` argument and check the effective sample at each point in time during the estimation. `"bootstrap_filter"` may not be that much cheaper in terms of computation time as we still have to evaluate g_t in Equation (16), (17), and (19) which is $\mathcal{O}(n_{\max}Nr)$ or $\mathcal{O}(n_{\max}N_S r)$ which is typically computationally expensive as $n_{\max}$ is large. On the other hand, the `"..._w_particles"` methods have a computational complexity of $\mathcal{O}(n_{\max}Nr^2)$ or $\mathcal{O}(n_{\max}N_S r^2)$ with a potentially much larger constant. Thus, the `"..._w_cloud_mean"` may be preferred.

The rest of the vignette covers the implemented methods. It is mainly included to show exactly what is computed and why. Further, I cover some currently not implemented extensions that may be implemented in the future.

1.3 Proposal Distributions and Re-sampling Weights

Algorithm 1 shows one of the particle smoothers shown by Fearnhead et al. (2010) in the first order state space model. In this situation $\mathbf{R} = \mathbf{I}_r$, $r = p$ and $\boldsymbol{\alpha}_t = \boldsymbol{\xi}_t$. We need to specify a series of proposal distributions and re-sampling weights. To show what is implemented and why, we first consider the model where

$$\boldsymbol{y}_t \mid \boldsymbol{\alpha}_t \sim \mathcal{N}(\mathbf{X}_t\boldsymbol{\alpha}_t + \boldsymbol{o}_t + \mathbf{Z}_t\boldsymbol{\omega}, \mathbf{H}_t)$$

for some known positive definite matrix $\mathbf{H}_t$. This is not implemented in this package but deriving optimal re-sampling weights and proposal distributions is possible in this case. In fact, it makes little sense to use a particle filter and particle smoother in this case since the

Kalman filter and an exact smoother can be applied. However, the results here will turn out to be useful to motivate the approximations we use later. The state space model is

$$\begin{aligned}
&\boldsymbol{y}_t \sim \mathcal{N}\left(\boldsymbol{\eta}_t, \mathbf{H}_t\right) & & \\
&\boldsymbol{\eta}_t = \mathbf{X}_t\boldsymbol{\alpha}_t + \boldsymbol{o}_t + \mathbf{Z}_t\boldsymbol{\omega} & & i = 1,\dots,n_t \\
&\boldsymbol{\alpha}_t = \mathbf{F}\boldsymbol{\alpha}_{t-1} + \mathbf{R}\boldsymbol{\epsilon}_t \qquad & \boldsymbol{\epsilon}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\right), & \quad t = 1,\dots,d \\
& & \boldsymbol{\alpha}_0 \sim \mathcal{N}\left(\boldsymbol{\mu}_0, \mathbf{Q}_0\right) &
\end{aligned}$$

We let $\boldsymbol{h}_t = \boldsymbol{o}_t + \mathbf{Z}_t\boldsymbol{\omega}$ such that $\boldsymbol{\eta}_t = \mathbf{X}_t\boldsymbol{\alpha}_t + \boldsymbol{h}_t$ to ease the notation. We first turn to the forward particle filter in Algorithm 2. Ideally, we want the re-sampling weights to be

$$\begin{aligned}
\beta_t^{(j)} &\propto p\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)} \\
&= \int g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) f\left(\boldsymbol{a}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) \mathrm{d}\boldsymbol{a}_t w_{t-1}^{(j)} \\
&= \phi\left(\boldsymbol{y}_t \middle| \mathbf{X}_t\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)} + \boldsymbol{h}_t, \mathbf{X}_t\mathbf{Q}\mathbf{X}_t^\top + \mathbf{H}_t\right) w_{t-1}^{(j)}
\end{aligned} \tag{2}$$

We can notice that setting $\beta_t^{(j)} = w_{t-1}^{(j)}$ yields the so-called sequential importance re-sampling algorithm. For the proposal distribution, the optimal proposal density is

$$q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) = p\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right)$$

where we find that

$$\begin{aligned}
\log p\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) &= \log p\left(\boldsymbol{\alpha}_t, \boldsymbol{y}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) + \dots \\
&= \log g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) + \log f\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) + \dots \\
&= -\frac{1}{2}\left(\boldsymbol{y}_t - \mathbf{X}_t\boldsymbol{\alpha}_t - \boldsymbol{h}_t\right)^\top \mathbf{H}_t^{-1}\left(\boldsymbol{y}_t - \mathbf{X}_t\boldsymbol{\alpha}_t - \boldsymbol{h}_t\right) \\
&\qquad - \frac{1}{2}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}\right)^\top \mathbf{Q}^{-1}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}\right) + \dots \\
&= -\frac{1}{2}\boldsymbol{\alpha}_t^\top\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\alpha}_t + \boldsymbol{\alpha}_t^\top\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}\left(\boldsymbol{\alpha}_{t-1}^{(j)}\right) + \dots
\end{aligned}$$

$$\boldsymbol{\Sigma}_t = \left(\mathbf{Q}^{-1} + \mathbf{X}_t^\top\mathbf{H}_t^{-1}\mathbf{X}_t\right)^{-1} \tag{3}$$

$$\boldsymbol{\mu}(\boldsymbol{x}) = \boldsymbol{\Sigma}_t\left(\mathbf{Q}^{-1}\mathbf{F}\boldsymbol{x} + \mathbf{X}_t^\top\mathbf{H}_t^{-1}(\boldsymbol{y}_t - \boldsymbol{h}_t)\right) \tag{4}$$

The $\dots$ are terms of the normalization constant. We recognize the multivariate normal distribution density and thus the optimal proposal density is

$$q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) = \phi\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\mu}(\boldsymbol{\alpha}_{t-1}^{(j)}), \boldsymbol{\Sigma}_t\right)$$

Alternatively, we can use the so-called bootstrap filter and let

$$q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) = \phi\left(\boldsymbol{\alpha}_t \middle| \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}, \mathbf{Q}\right) \tag{5}$$

which we can sample from in $\mathcal{O}\left(Np^2\right)$ time if we have a pre-computed Cholesky decomposition of $\mathbf{Q}$. This is computationally cheap compared to optimal solution which is $\mathcal{O}\left(Np^2 + p^3 + n_{\max}p^2\right)$ but it is not optimal.

Backward filter (Algorithm 3)

We need to specify the artificial prior $\gamma_t\left(\boldsymbol{\alpha}_t\right)$ for our artificial backward distribution. Briers et al. (2009, p. 69-70) provides recommendation on the selection. One suggestion is the artificial density function

$$
\begin{aligned}
\gamma_t\left(\boldsymbol{\alpha}_t\right) &= \phi\left(\boldsymbol{\alpha}_t \middle| \overleftarrow{\boldsymbol{m}}_t, \overleftarrow{\mathbf{P}}_t\right) \\
\overleftarrow{\boldsymbol{m}}_t &= \mathbf{F}^t \boldsymbol{\mu}_0 \\
\overleftarrow{\mathbf{P}}_t &= \begin{cases} \mathbf{Q}_0 & t = 0 \\ \mathbf{F}\overleftarrow{\mathbf{P}}_{t-1}\mathbf{F}^\top + \mathbf{Q} & t > 0 \end{cases}
\end{aligned} \tag{6}
$$

The backward arrows are added to stress that these are means and covariance matrices which we use in the artificial distribution we target in the backward particle filter. The artificial distribution we target in backward particle filters has the following conditional density functions

$$
\begin{aligned}
\widetilde{p}\left(\boldsymbol{\alpha}_{t:d} \middle| \boldsymbol{y}_{t:d}\right) &\propto \gamma_t\left(\boldsymbol{\alpha}_t\right) \prod_{i=t}^{d} g_i\left(\boldsymbol{y}_i \middle| \boldsymbol{\alpha}_i\right) \prod_{i=t}^{d-1} f\left(\boldsymbol{\alpha}_{i+1} \middle| \boldsymbol{\alpha}_i\right) \\
\widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{(t+1):d}\right) &\propto \gamma_t(\boldsymbol{\alpha}_t) \int \widetilde{p}\left(\boldsymbol{a}_{t+1} \middle| \boldsymbol{y}_{(t+1):d}\right) \frac{f\left(\boldsymbol{a}_{t+1} \middle| \boldsymbol{\alpha}_t\right)}{\gamma_{t+1}(\boldsymbol{a}_{t+1})} \mathrm{d}\boldsymbol{a}_{t+1} \\
\widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{t:d}\right) &\propto g_t\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_t\right) \widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{(t+1):d}\right) \\
\widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t+1}\right) &= \frac{f\left(\boldsymbol{\alpha}_{t+1} \middle| \boldsymbol{\alpha}_t\right) \gamma_t(\boldsymbol{\alpha}_t)}{\gamma_{t+1}(\boldsymbol{\alpha}_{t+1})}
\end{aligned} \tag{7}
$$

where we have left out some of the normalization constants. Sampling from this artificial distribution turns out to be useful as it gives us an approximation of a conditional density we need up to a constant (see Section 6). To derive the re-sampling weight, we first find an expression for the density $\widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t+1}\right)$. We can observe that

$$
\begin{aligned}
\log \widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t+1}\right) &= \log f\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t+1}\right) + \log \gamma_t(\boldsymbol{\alpha}_t) + \dots \\
&= -\frac{1}{2}\boldsymbol{\alpha}_t^\top \overleftarrow{\mathbf{S}}_t^{-1} \boldsymbol{\alpha}_t - \boldsymbol{\alpha}_t^\top \overleftarrow{\mathbf{S}}_t^{-1} \overleftarrow{\boldsymbol{a}}_t(\boldsymbol{\alpha}_{t+1}) + \dots \\
\overleftarrow{\mathbf{S}}_t &= \left(\mathbf{P}_t^{-1} + \mathbf{F}^\top \mathbf{Q}^{-1} \mathbf{F}\right)^{-1} \\
\overleftarrow{\boldsymbol{a}}_t(\boldsymbol{x}) &= \overleftarrow{\mathbf{S}}_t \left(\mathbf{P}_t^{-1} \boldsymbol{m}_t + \mathbf{F}^\top \mathbf{Q}^{-1} \boldsymbol{x}\right)
\end{aligned}
$$

so

$$
\widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t+1}\right) = \phi\left(\boldsymbol{\alpha}_t \middle| \overleftarrow{\boldsymbol{a}}_t(\boldsymbol{\alpha}_{t+1}), \overleftarrow{\mathbf{S}}_t\right) \tag{8}
$$

.

As shown by Fearnhead et al. (2010), we can show that

$$
\begin{aligned}
\overleftarrow{\mathbf{S}}_t &= \overleftarrow{\mathbf{P}}_t \mathbf{F}^\top \overleftarrow{\mathbf{P}}_{t+1}^{-1} \mathbf{Q} \mathbf{F}^{-\top} \\
\overleftarrow{\boldsymbol{a}}_t(\boldsymbol{x}) &= \overleftarrow{\mathbf{P}}_t \mathbf{F}^\top \overleftarrow{\mathbf{P}}_{t+1}^{-1} \boldsymbol{x} + \overleftarrow{\mathbf{S}}_t \overleftarrow{\mathbf{P}}_t^{-1} \overleftarrow{\boldsymbol{m}}_t
\end{aligned} \tag{9}
$$

e.g., by

$$
\begin{aligned}
&\left(\overleftarrow{\mathbf{P}}_t\mathbf{F}^\top\overleftarrow{\mathbf{P}}_{t+1}^{-1}\mathbf{Q}\mathbf{F}^{-\top}\right)^{-1}\left(\mathbf{P}_t^{-1}+\mathbf{F}^\top\mathbf{Q}^{-1}\mathbf{F}\right)^{-1}\\
&\qquad=\mathbf{F}^\top\mathbf{Q}^{-1}\overleftarrow{\mathbf{P}}_{t+1}\mathbf{F}^{-\top}\overleftarrow{\mathbf{P}}_t^{-1}\left(\mathbf{P}_t^{-1}+\mathbf{F}^\top\mathbf{Q}\mathbf{F}\right)^{-1}\\
&\qquad=\mathbf{F}^\top\mathbf{Q}^{-1}\overleftarrow{\mathbf{P}}_{t+1}\mathbf{F}^{-\top}\overleftarrow{\mathbf{P}}_t^{-1}\left(\overleftarrow{\mathbf{P}}_t-\overleftarrow{\mathbf{P}}_t\mathbf{F}^\top\left(\mathbf{Q}+\mathbf{F}\overleftarrow{\mathbf{P}}_t\mathbf{F}^\top\right)^{-1}\mathbf{F}\overleftarrow{\mathbf{P}}_t\right)\\
&\qquad=\mathbf{F}^\top\mathbf{Q}^{-1}\overleftarrow{\mathbf{P}}_{t+1}\mathbf{F}^{-\top}\left(\mathbf{I}-\mathbf{F}^\top\overleftarrow{\mathbf{P}}_{t+1}^{-1}\mathbf{F}\overleftarrow{\mathbf{P}}_t\right)\\
&\qquad=\mathbf{F}^\top\mathbf{Q}^{-1}\overleftarrow{\mathbf{P}}_{t+1}\mathbf{F}^{-\top}-\mathbf{F}^\top\mathbf{Q}^{-1}\mathbf{F}\overleftarrow{\mathbf{P}}_t\\
&\qquad=\mathbf{F}^\top\mathbf{Q}^{-1}\left(\mathbf{F}\overleftarrow{\mathbf{P}}_t\mathbf{F}^\top+\mathbf{Q}\right)\mathbf{F}^{-\top}-\mathbf{F}^\top\mathbf{Q}^{-1}\mathbf{F}\overleftarrow{\mathbf{P}}_t\\
&\qquad=\mathbf{I}
\end{aligned}
$$

where we assume that all matrices are non-singular and we use the Woodbury matrix identity. Similar arguments can be used for $\overleftarrow{\boldsymbol{a}}_t(\boldsymbol{x})$. Using the above, we can find that the optimal re-sampling weights are

$$
\begin{aligned}
\widetilde{\beta}_t^{(k)}&\propto\widetilde{p}\left(\boldsymbol{y}_t\middle|\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)\widetilde{w}_{t+1}^{(k)}\\
&\propto\int g_t\left(\boldsymbol{y}_i|\boldsymbol{a}_t\right)\widetilde{p}\left(\boldsymbol{a}_t\middle|\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)\mathrm{d}\boldsymbol{a}_t\widetilde{w}_{t+1}^{(k)}\\
&=\phi\left(\boldsymbol{y}_t\middle|\mathbf{X}_t\overleftarrow{\boldsymbol{a}}_t(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)})+\boldsymbol{h}_t,\mathbf{X}_t\overleftarrow{\mathbf{S}}_t\mathbf{X}_t^\top+\mathbf{H}_t\right)\widetilde{w}_{t+1}^{(k)}
\end{aligned}
\tag{10}
$$

or we can set the re-sampling weights proportional to $\widetilde{\beta}_t^{(k)}\propto\widetilde{w}_{t+1}^{(k)}$ and get a sequential importance re-sampling like algorithm. As for the proposal distribution, the optimal density is

$$
\begin{aligned}
\log\widetilde{q}\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{y}_t,\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)&=\log\gamma_t\left(\boldsymbol{\alpha}_t\right)+\log g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right)+\log f\left(\boldsymbol{\alpha}_{t+1}^{(k)}\middle|\boldsymbol{\alpha}_t\right)+\dots\\
&=-\frac{1}{2}\boldsymbol{\alpha}_t^\top\overleftarrow{\mathbf{\Sigma}}_t^{-1}\boldsymbol{\alpha}_t+\boldsymbol{\alpha}_t^\top\overleftarrow{\mathbf{\Sigma}}_t^{-1}\overleftarrow{\boldsymbol{\mu}}(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)})+\dots\\
\overleftarrow{\mathbf{\Sigma}}_t&=\left(\mathbf{P}_t^{-1}+\mathbf{F}^\top\mathbf{Q}^{-1}\mathbf{F}+\mathbf{X}_t^\top\mathbf{H}_t^{-1}\mathbf{X}_t\right)^{-1}\\
\overleftarrow{\boldsymbol{\mu}}_t(\boldsymbol{x})&=\mathbf{\Sigma}_t\left(\mathbf{P}_t^{-1}\boldsymbol{m}_t+\mathbf{F}^\top\mathbf{Q}^{-1}\boldsymbol{x}+\mathbf{X}_t^\top\mathbf{H}_t^{-1}(\boldsymbol{y}_t-\boldsymbol{h}_t)\right)
\end{aligned}
$$

Thus, we set

$$
\widetilde{q}\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{y}_t,\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)=\phi\left(\boldsymbol{\alpha}_t\middle|\overleftarrow{\boldsymbol{\mu}}_t(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}),\overleftarrow{\mathbf{\Sigma}}_t\right)
$$

A computationally simpler but non-optimal option is to use a method like the bootstrap filter and set

$$
\widetilde{q}\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{y}_t,\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)=\widetilde{p}\left(\boldsymbol{\alpha}_t\middle|\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)\tag{11}
$$

Combining / smoothing (Algorithm 1)

We end this example with the conditional Gaussian observable outcome model with the proposal distribution needed for Algorithm 1. We want to select

$$
\begin{aligned}
q\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)},\boldsymbol{y}_t,\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)&=p\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)},\boldsymbol{y}_t,\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)\\
&\propto g\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right)f\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)}\right)f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\middle|\boldsymbol{\alpha}_t\right)
\end{aligned}
$$

Looking at the log density as we did before, we find that

$$\begin{aligned}
&\log q\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)},\boldsymbol{y}_t,\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)\\
&\qquad=\log g\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right)+\log f\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)}\right)+\log f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\middle|\boldsymbol{\alpha}_t\right)+\dots\\
&\qquad=-\frac{1}{2}\boldsymbol{\alpha}_t^\top\overleftrightarrow{\boldsymbol{\Sigma}}_t^{-1}\boldsymbol{\alpha}_t+\boldsymbol{\alpha}_t^\top\overleftrightarrow{\boldsymbol{\Sigma}}_t^{-1}\overleftrightarrow{\boldsymbol{\mu}}(\boldsymbol{\alpha}_{t-1}^{(j)},\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)})+\dots
\end{aligned}$$

$$\overleftrightarrow{\boldsymbol{\Sigma}}_t=\left(\mathbf{Q}^{-1}+\mathbf{F}^\top\mathbf{Q}^{-1}\mathbf{F}+\mathbf{X}_t^\top\mathbf{H}^{-1}\mathbf{X}_t\right)^{-1} \tag{12}$$

$$\overleftrightarrow{\boldsymbol{\mu}}_t(\boldsymbol{x},\overleftarrow{\boldsymbol{x}})=\boldsymbol{\Sigma}_t\left(\mathbf{Q}^{-1}\mathbf{F}\boldsymbol{x}+\mathbf{F}^\top\mathbf{Q}^{-1}\overleftarrow{\boldsymbol{x}}+\mathbf{X}_t^\top\mathbf{H}_t^{-1}(\boldsymbol{y}_t-\boldsymbol{h}_t)\right) \tag{13}$$

so that

$$q\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)},\boldsymbol{y}_t,\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)=\phi\left(\boldsymbol{\alpha}_t\middle|\overleftrightarrow{\boldsymbol{\mu}}_t(\boldsymbol{\alpha}_{t-1}^{(j)},\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}),\boldsymbol{\Sigma}_t\right)$$

Alternatively, we can use a method like the bootstrap filter with a proposal distribution with

$$\begin{aligned}
\overleftrightarrow{\boldsymbol{\Sigma}}_t&=\left(\mathbf{Q}^{-1}+\mathbf{F}^\top\mathbf{Q}^{-1}\mathbf{F}\right)^{-1}\\
\overleftrightarrow{\boldsymbol{\mu}}_t(\boldsymbol{x},\overleftarrow{\boldsymbol{x}})&=\boldsymbol{\Sigma}_t\left(\mathbf{Q}^{-1}\mathbf{F}\boldsymbol{x}+\mathbf{F}^\top\mathbf{Q}^{-1}\overleftarrow{\boldsymbol{x}}\right)
\end{aligned} \tag{14}$$

This is not optimal but faster.

Algorithm 1 $\mathcal{O}(N)$ particle smoother using the method in Fearnhead et al. (2010).

Input:

$\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{a}_0, \mathbf{X}_1, \dots, \mathbf{X}_d, \mathbf{Z}_1, \dots, \mathbf{Z}_d, \boldsymbol{o}_1, \dots, \boldsymbol{o}_d, \boldsymbol{y}_1, \dots, \boldsymbol{y}_d, R_1, \dots, R_d, \boldsymbol{\omega}$

Proposal distribution with density

$$q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \tag{15}$$

procedure Filter forward

Run a forward particle filter to get particle clouds $\left\{\boldsymbol{\alpha}_t^{(j)}, w_t^{(j)}, \beta_{t+1}^{(j)}\right\}_{j=1,\dots,N}$ approximating the density $p\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{1:t}\right)$ for $t = 0, 1, \dots, d$. See Algorithm 2.

procedure Filter backwards

Run a similar backward filter to get $\left\{\widetilde{\boldsymbol{\alpha}}_t^{(k)}, \widetilde{w}_t^{(k)}, \widetilde{\beta}_{t-1}^{(k)}\right\}_{k=1,\dots,N}$ approximating the artificial density $\widetilde{p}\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{t:d}\right)$ for $t = d+1, d, d-1, \dots, 1$. See Algorithm 3.

procedure Smooth (combine)

for $t = 1, \dots, d$ **do**

Re-sample

Sample $i = 1, 2, \dots, N_s$ pairs of $(j_i, k_i) \in N^2$ where each component is independently sampled using re-sampling weights $\beta_t^{(j)}$ and $\widetilde{\beta}_t^{(k)}$.

Propagate

Sample particles $\widehat{\boldsymbol{\alpha}}_t^{(i)}$ from the proposal distribution $\widetilde{q}\left(\cdot \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right)$.

Re-weight

Assign each particle a weight

$$\widehat{w}_t^{(i)} \propto \frac{f\left(\widehat{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}\right) g_t\left(\boldsymbol{y}_t \middle| \widehat{\boldsymbol{\alpha}}_t^{(i)}\right) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)} \middle| \widehat{\boldsymbol{\alpha}}_t^{(i)}\right) w_{t-1}^{(j_i)} \widetilde{w}_{t+1}^{(k_i)}}{\widetilde{q}\left(\widehat{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right) \beta_t^{(j_i)} \widetilde{\beta}_t^{(k_i)} \gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right)} \tag{16}$$

Algorithm 2 Forward filter as in Pitt and Shephard (1999). It is equivalent with Doucet and Johansen (2009, p. 20 and 25). The version and notation below is from Fearnhead et al. (2010, p. 449).

Input:

Proposal distribution with density

$$q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right)$$

Function h to compute re-sampling weights

$$\beta_t^{(j)} \propto h(\boldsymbol{y}_t, \boldsymbol{\alpha}_{t-1}^{(j)}) w_{t-1}^{(j)}$$

Sample $\boldsymbol{\alpha}_0^{(1)}, \dots, \boldsymbol{\alpha}_0^{(N_f)}$ particles from $\mathcal{N}(\boldsymbol{\mu}_0, \mathbf{Q}_0)$ and set the weights $w_0^{(1)}, \dots, w_0^{(N_f)}$ to $1/N_f$.

for $t = 1, \dots, d$ **do**

  procedure Re-sample

    Compute re-sampling weights $\beta_t^{(j)}$ using h and re-sample according to $\beta_t^{(j)}$ to get indices $j_1, \dots j_N$. If we do not re-sample then set $\beta_t^{(j)} = 1/N$ or $1/N_f$ at time $t = 1$.

  procedure Propagate

    Sample new particles $\boldsymbol{\alpha}_t^{(i)}$ using the proposal distribution $q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t\right)$.

  procedure Re-weight

    Re-weight particles using

$$w_t^{(i)} \propto \frac{g_t\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_t^{(i)}\right) f\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}\right) w_{t-1}^{(j_i)}}{q\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t\right) \beta_t^{(j_i)}} \tag{17}$$

Algorithm 3 Backwards filter. See Briers et al. (2009) and Fearnhead et al. (2010).

Input:

An artificial distribution

$$\widetilde{p}\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{t:d}\right) \propto \gamma_t\left(\boldsymbol{\alpha}_t\right) p\left(\boldsymbol{y}_{t:d}|\boldsymbol{\alpha}_t\right) \tag{18}$$

with an artificial prior distribution $\gamma_t\left(\boldsymbol{\alpha}_t\right)$.

Proposal distribution

$$\widetilde{q}\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)$$

Function h to compute re-sampling weights

$$\widetilde{\beta}_t^{(k)} \propto h(\boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)})\widetilde{w}_{t+1}^{(k)}$$

Sample $\widetilde{\boldsymbol{\alpha}}_{d+1}^{(1)}, \dots, \widetilde{\boldsymbol{\alpha}}_{d+1}^{(N_f)}$ particles from $\gamma_{d+1}(\cdot)$ and set the weights $\widetilde{w}_{d+1}^{(1)}, \dots, w_{d+1}^{(N_f)}$ to $1/N_f$.

for $t = d, \dots, 1$ **do**

  procedure RE-SAMPLE

    Compute re-sampling weights $\widetilde{\beta}_t^{(k)}$ using h and re-sample according to $\widetilde{\beta}_t^{(k)}$ to get indices $k_1, \dots k_N$. If we do not re-sample then set $\widetilde{\beta}_t^{(k)} = 1/N$ or $1/N_f$ at time $t = d$.

  procedure PROPAGATE

    Sample new particles $\widetilde{\boldsymbol{\alpha}}_t^{(i)}$ using the proposal distribution $\widetilde{q}\left(\boldsymbol{\alpha}_t\middle|\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}, \boldsymbol{y}_t\right)$.

  procedure RE-WEIGHT

    Re-weight particles using

$$\widetilde{w}_t^{(i)} \propto \frac{g_t\left(\boldsymbol{y}_t\middle|\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\middle|\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right) \gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right) \widetilde{w}_{t+1}^{(k_i)}}{q\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\middle|\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}, \boldsymbol{y}_t\right) \gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right) \beta_t^{(k_i)}} \tag{19}$$

2 Non-linear Conditional Observation Model

If we go back to the model in Equation (1) then $\boldsymbol{y}_t \mid \boldsymbol{\alpha}_t$ is not a multivariate normal distribution for the implemented models. In this case, we have no closed from solutions for the optimal re-sampling weights, and we do not know the following conditional distributions: $\boldsymbol{\alpha}_t \mid \boldsymbol{y}_t, \boldsymbol{\alpha}_{t-1}$, $\boldsymbol{\alpha}_t \mid \boldsymbol{y}_t, \boldsymbol{\alpha}_{t+1}$ (in the artificial distribution $\widetilde{\mathrm{P}}$), and $\boldsymbol{\alpha}_t \mid \boldsymbol{y}_t, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\alpha}_{t+1}$. However, assume that $g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right)$ is log-concave in $\boldsymbol{\alpha}_t$. If this is true then it is easy to show that all of the previous three conditional distributions are unimodal. Hence, we can make a multivariate normal approximation as in Pitt and Shephard (1999). To do so, we make a second order Taylor expansion around some value $\boldsymbol{z}$ to get

$$
\begin{aligned}
k_t(\boldsymbol{\alpha}_t) &= \log g_t\left(\boldsymbol{y}_t|\boldsymbol{\eta}(\boldsymbol{\alpha}_t)\right), \qquad \boldsymbol{\eta}(\boldsymbol{\alpha}_t) = \mathbf{X}_t\boldsymbol{\alpha}_t + \boldsymbol{h}_t \\
\log g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) &\approx \mathrm{D}k_t(\boldsymbol{z})(\boldsymbol{\alpha}_t - \boldsymbol{z}) + \frac{1}{2}(\boldsymbol{\alpha}_t - \boldsymbol{z})^\top \mathrm{H}k_t(\boldsymbol{z})(\boldsymbol{\alpha}_t - \boldsymbol{z}) + \dots \\
&= \boldsymbol{\alpha}_t^\top \mathrm{D}k_t(\boldsymbol{z})^\top - \frac{1}{2}(\boldsymbol{\alpha}_t - \boldsymbol{z})^\top\left(-\mathrm{H}k_t(\boldsymbol{z})\right)(\boldsymbol{\alpha}_t - \boldsymbol{z}) + \dots \\
&= \boldsymbol{\alpha}_t^\top\left(-\mathrm{H}k_t(\boldsymbol{z})\right)\left(\boldsymbol{z} - \mathrm{H}k_t(\boldsymbol{z})^{-1}\mathrm{D}k_t(\boldsymbol{z})^\top\right) - \frac{1}{2}\boldsymbol{\alpha}_t^\top\left(-\mathrm{H}k_t(\boldsymbol{z})\right)\boldsymbol{\alpha}_t + \dots
\end{aligned}
$$

where $\dots$ includes the zero order term, $\mathrm{D}k_t$ is the Jacobian, and $\mathrm{H}k_t$ denotes the Hessian. I add a subscript to D and H to which variable the Jacobian or Hessian is with respect to. We still assume that we use a first order state space model such that $r = p$. We notice that

$$
\begin{aligned}
\mathrm{H}k_t(\boldsymbol{z}) &= \mathrm{D}_{\boldsymbol{\alpha}_t}\boldsymbol{\eta}(\boldsymbol{z})^\top \mathrm{H}_{\boldsymbol{\eta}} \log g_t\left(\boldsymbol{y}_t|\boldsymbol{\eta}(\boldsymbol{z})\right)\mathrm{D}_{\boldsymbol{\alpha}_t}\boldsymbol{\eta}(\boldsymbol{z}) \\
&= \mathbf{X}_t^\top(-\mathbf{G}_t(\boldsymbol{z}))\mathbf{X}_t, \qquad \mathbf{G}_t(\boldsymbol{z}) = -\mathrm{H}_{\boldsymbol{\eta}}\log g_t\left(\boldsymbol{y}_t|\boldsymbol{\eta}(\boldsymbol{z})\right)
\end{aligned}
$$

which follows from the chain rule and we use that $\mathrm{H}_{\boldsymbol{\alpha}_t}\boldsymbol{\eta}(\boldsymbol{z}) = \mathbf{0}$. Thus,

$$
\begin{aligned}
\log g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) &\approx \boldsymbol{\alpha}_t^\top\mathbf{X}_t^\top\mathbf{G}_t(\boldsymbol{z})\boldsymbol{u}_t(\boldsymbol{z}) - \frac{1}{2}\boldsymbol{\alpha}_t\mathbf{X}_t^\top\mathbf{G}_t(\boldsymbol{z})\mathbf{X}_t\boldsymbol{\alpha}_t \\
\boldsymbol{u}_t(\boldsymbol{z}) &= \mathbf{X}_t\boldsymbol{z} - \mathbf{X}_t\mathrm{H}k_t(\boldsymbol{z})^{-1}\mathrm{D}k_t(\boldsymbol{z})^\top
\end{aligned}
$$

This yields the following multivariate normal approximation

$$
g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) \approx \phi\left(\mathbf{X}_t\boldsymbol{\alpha}_t \middle| \boldsymbol{u}_t(\boldsymbol{z}), \mathbf{G}_t(\boldsymbol{z})^{-1}\right)
$$

The Taylor approximation is easily used in the proposal distributions. E.g., for given $\boldsymbol{z}$, we get the following mean and covariance matrix analogues to Equation (3) and (4) in the proposal distribution in the forward particle filter

$$
\begin{aligned}
\boldsymbol{\Sigma}_t(\boldsymbol{z}) &= \left(\mathbf{Q}^{-1} + \mathbf{X}_t^\top\mathbf{G}_t(\boldsymbol{z})\mathbf{X}_t\right)^{-1} \\
\boldsymbol{\mu}_t(\boldsymbol{x}, \boldsymbol{z}) &= \boldsymbol{\Sigma}_t(\boldsymbol{z})\left(\mathbf{Q}^{-1}\mathbf{F}\boldsymbol{x} + \mathbf{X}_t^\top\mathbf{G}_t(\boldsymbol{z})\boldsymbol{u}_t(\boldsymbol{z})\right)
\end{aligned}
$$

As for the re-sampling weights, we can use

$$\begin{aligned}
\widehat{\boldsymbol{\alpha}} &= \boldsymbol{\mu}_t(\boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{z}) \\
\beta_t^{(j)} &\propto p\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)} \\
&= \frac{p\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)}{g_t\left(\boldsymbol{y}_t|\widehat{\boldsymbol{\alpha}}\right) f\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)} g_t\left(\boldsymbol{y}_t|\widehat{\boldsymbol{\alpha}}\right) f\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)} \\
&= \frac{g_t\left(\boldsymbol{y}_t|\widehat{\boldsymbol{\alpha}}\right) f\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)}}{p\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right)} \\
&\approx \frac{g_t\left(\boldsymbol{y}_t|\widehat{\boldsymbol{\alpha}}\right) f\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)}}{q\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right)}
\end{aligned}$$

as in Fearnhead et al. (2010). We can approximate the backwards particle filter re-sampling weights in equation (10) in a similar way

$$\begin{aligned}
\widetilde{\beta}_t^{(k)} &\propto \widetilde{p}\left(\boldsymbol{y}_t \middle| \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \widetilde{w}_{t+1}^{(k)} \\
&\approx \frac{g_t\left(\boldsymbol{y}_t|\widehat{\boldsymbol{\alpha}}\right) \widetilde{p}\left(\widehat{\boldsymbol{\alpha}} \middle| \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \widetilde{w}_{t+1}^{(k)}}{\widetilde{q}\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)} \\
&= \frac{g_t\left(\boldsymbol{y}_t|\widehat{\boldsymbol{\alpha}}\right) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \widehat{\boldsymbol{\alpha}}\right) \gamma_t(\widehat{\boldsymbol{\alpha}}) \widetilde{w}_{t+1}^{(k)}}{\widetilde{q}\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \gamma_{t+1}(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)})} && (20) \\
\widehat{\boldsymbol{\alpha}} &= \overleftarrow{\boldsymbol{\mu}}(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}, \boldsymbol{z}) \\
\overleftarrow{\boldsymbol{\mu}}(\boldsymbol{x}, \boldsymbol{z}) &= \overleftarrow{\boldsymbol{\Sigma}}_t(\boldsymbol{z}) \left(\mathbf{P}_t^{-1}\boldsymbol{m}_t + \mathbf{F}^\top \mathbf{Q}^{-1}\boldsymbol{x} + \mathbf{X}_t^\top \mathbf{G}_t(\boldsymbol{z})\boldsymbol{u}_t(\boldsymbol{z})\right) && (21) \\
\overleftarrow{\boldsymbol{\Sigma}}_t(\boldsymbol{z}) &= \left(\mathbf{P}_t^{-1} + \mathbf{F}^\top \mathbf{Q}^{-1}\mathbf{F} + \mathbf{X}_t^\top \mathbf{G}_t(\boldsymbol{z})\mathbf{X}_t\right)^{-1} && (22)
\end{aligned}$$

We may also use a multivariate t-distribution for the proposal distribution to get heavier tails than we do with the multivariate normal distribution. This is important as too light tailed proposal distributions (relative to the target) can yield few large importance weights.

2.1 Where to Make the Expansion

An options is to make the Taylor expansion at a mode for each particle or particle pair in the smoothing step. This yields

$$\begin{aligned}
\boldsymbol{z}^{(j)} &= \arg\max_{\boldsymbol{z}} g_t\left(\boldsymbol{y}_t|\boldsymbol{z}\right) f\left(\boldsymbol{z} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) \\
\boldsymbol{z}^{(k)} &= \arg\max_{\boldsymbol{z}} g_t\left(\boldsymbol{y}_t|\boldsymbol{z}\right) \gamma_t(\boldsymbol{z}) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \boldsymbol{z}\right) \\
\boldsymbol{z}^{(i)} &= \arg\max_{\boldsymbol{z}} f\left(\boldsymbol{z} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}\right) g_t\left(\boldsymbol{y}_t|\boldsymbol{z}\right) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)} \middle| \boldsymbol{z}\right)
\end{aligned}$$

for respectively the forward particle filter, backward particle filter, and smoother. The downside is a $\mathcal{O}\left(r^2 n_{\max} N_S\right)$ or $\mathcal{O}\left(r^2 n_{\max} N\right)$ computational complexity at each time point as we have to evaluate $\mathbf{X}_t^\top \mathbf{G}_t(\boldsymbol{z})\mathbf{X}_t$ for each particle or particle pair. Instead we can make the approximation once at each time point at respectively $\sum_{i=1}^N w_{t-1}^{(j)} \boldsymbol{\alpha}_{t-1}^{(j)}$, $\sum_{i=1}^N \widetilde{w}_{t+1}^{(j)} \widetilde{\boldsymbol{\alpha}}_{t+1}^{(j)}$, and

$$\left(\mathbf{Q}^{-1} + \mathbf{F}^\top \mathbf{Q}^{-1}\mathbf{F}\right)^{-1} \left(\mathbf{Q}^{-1}\mathbf{F}\sum_{i=1}^N w_{t-1}^{(j)} \boldsymbol{\alpha}_{t-1}^{(j)} + \mathbf{F}^\top \mathbf{Q}^{-1} \sum_{i=1}^N \widetilde{w}_{t+1}^{(j)} \widetilde{\boldsymbol{\alpha}}_{t+1}^{(j)}\right)$$

which will reduce the computational complexity at each time point to $\mathcal{O}\left(rn_{\max}N_S + rpn_{\max}\right)$ or $\mathcal{O}\left(rn_{\max}N + rpn_{\max}\right)$.

3 Higher Order State Space Models

Now, we will consider the case where $p > r$. Currently this is not supported in the package but may be in the future. An example is a seconder order vector autoregression, VAR$\,(2)$, with $2r = p$ and

$$\begin{aligned} \mathbf{F} &= \begin{pmatrix} \mathbf{F}_1 & \mathbf{F}_2 \\ \mathbf{I}_r & \mathbf{0} \end{pmatrix}, \quad \mathbf{F}_i \in \mathbb{R}^{r\times r} \\ \boldsymbol{\alpha}_t &= \begin{pmatrix} \boldsymbol{\xi}_t^\top & \boldsymbol{\xi}_{t-1}^\top \end{pmatrix}^\top \end{aligned}$$

Here

$$\begin{aligned} f\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}\right) &= \delta_{\mathbf{KF}\boldsymbol{\alpha}_{t-1}}\left(\mathbf{K}\boldsymbol{\alpha}_t\right)\phi\left(\mathbf{R}^+\boldsymbol{\alpha}_t \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}, \mathbf{Q}\right), \qquad \mathbf{K} = \begin{pmatrix} \mathbf{0} \\ \mathbf{I}_{p-r}\end{pmatrix} \\ &= \delta_{\boldsymbol{\xi}_{t-1}}\left(\mathbf{K}\boldsymbol{\alpha}_t\right)\phi\left(\boldsymbol{\xi}_t \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}, \mathbf{Q}\right) \end{aligned}$$

the second equality follows in a seconder order vector autoregression. This is easily implemented in the forward particle filter by sampling $\boldsymbol{\xi}_t \mid \boldsymbol{\alpha}_{t-1}$ and setting the remaining $p-r$ variables of $\boldsymbol{\alpha}_t$ to $\mathbf{KF}\boldsymbol{\alpha}_{t-1}$ (the last $p-r$ rows of $\mathbf{F}\boldsymbol{\alpha}_{t-1}$). It is not as easy for the backward filter. To see this, consider the artificial transition density in Equation (7)

$$\begin{aligned} \widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t+1}\right) &= \frac{f\left(\boldsymbol{\alpha}_{t+1} \middle| \boldsymbol{\alpha}_t\right)\gamma_t(\boldsymbol{\alpha}_t)}{\gamma_{t+1}(\boldsymbol{\alpha}_{t+1})} \\ &= \frac{\delta_{\mathbf{KF}\boldsymbol{\alpha}_t}\left(\mathbf{K}\boldsymbol{\alpha}_{t+1}\right)\phi\left(\mathbf{R}^+\boldsymbol{\alpha}_{t+1} \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_t, \mathbf{Q}\right)\gamma_t(\boldsymbol{\alpha}_t)}{\gamma_{t+1}(\boldsymbol{\alpha}_{t+1})} \end{aligned}$$

where the last equality holds in this example. This distribution is obviously degenerate as it only has mass at the point $\boldsymbol{\xi}_t = \mathbf{KF}\boldsymbol{\alpha}_t$ in the VAR(2) model.

3.1 Backward Particle Filter

What we can do in the backward particle filter in a VAR$\,(o)$ model with $o > 1$ is to sample $\boldsymbol{\xi}_t$ at time t given $\widetilde{\boldsymbol{\alpha}}_{t+o}^{(k)}$. Thus, we start Algorithm 3 by sampling $\boldsymbol{\alpha}_{d+o}$. Further, we change the artificial prior distribution density in Equation (6) to

$$\begin{aligned}
\gamma_t\left(\boldsymbol{\alpha}_t\right) &= \phi\left(\boldsymbol{\alpha}_t \middle| \overleftarrow{\boldsymbol{m}}_t, \overleftarrow{\mathbf{P}}_t\right) \\
\gamma_t\left(\boldsymbol{\xi}_t\right) &= \phi\left(\boldsymbol{\alpha}_t \middle| \mathbf{R}^+\overleftarrow{\boldsymbol{m}}_t, \mathbf{R}^+\overleftarrow{\mathbf{P}}_t\mathbf{R}^{+\top}\right) \\
\overleftarrow{\boldsymbol{m}}_t &= \mathbf{F}^t\boldsymbol{\mu}_0 \\
\overleftarrow{\mathbf{P}}_t &= \begin{cases} \mathbf{Q}_0 & t = 0 \\ \mathbf{F}\overleftarrow{\mathbf{P}}_{t-1}\mathbf{F}^\top + \mathbf{R}\mathbf{Q}\mathbf{R}^\top & t > 0 \end{cases}
\end{aligned}$$

Next, we find the conditional distribution $\widetilde{\boldsymbol{\alpha}}_{t+o}^{(k)} \mid \boldsymbol{\xi}_t$. To do so, we first find the join distribution of $(\boldsymbol{\alpha}_{t+o}^\top, \boldsymbol{\xi}_t^\top)^\top$. We can find that

$$\begin{aligned}
\boldsymbol{\xi}_k &= \mathbf{R}^+\mathbf{F}^k\boldsymbol{\alpha}_0 + \sum_{i=1}^{k}\mathbf{G}(k-i)\boldsymbol{\epsilon}_i \\
\mathbf{G}(j) &= \begin{cases} \mathbf{I}_r & j = 0 \\ \sum_{i=1}^{\min(j,o)}\mathbf{F}_i\mathbf{G}(j-i) & j > 0 \end{cases}
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathrm{E}\left(\boldsymbol{\xi}_t\right) &= \mathbf{R}^+\mathbf{F}^k\boldsymbol{\mu}_0 = \mathbf{R}^+\boldsymbol{m}_t \\
\mathrm{E}\left(\boldsymbol{\alpha}_t\right) &= \boldsymbol{m}_t \\
\mathrm{Var}\left(\boldsymbol{\xi}_t\right) &= \sum_{i=1}^{t}\mathbf{G}(t-i)\mathbf{Q}\mathbf{G}(t-i)^\top + \mathbf{R}^+\mathbf{F}^t\mathbf{Q}_0\mathbf{F}^{t\top}\mathbf{R}^{+\top} \\
&= \mathbf{R}^+\mathbf{P}_t\mathbf{R}^{+\top} \\
\mathrm{Cov}\left(\boldsymbol{\xi}_k, \boldsymbol{\xi}_l\right) &= \sum_{i=1}^{l}\mathbf{G}(k-i)\mathbf{Q}\mathbf{G}(l-i)^\top + \mathbf{R}^+\mathbf{F}^k\mathbf{Q}_0\mathbf{F}^{l\top}\mathbf{R}^{+\top}, \qquad k > l \\
\mathrm{Var}\left(\boldsymbol{\alpha}_t\right) &= \mathbf{P}_t
\end{aligned}$$

with which we can find that the joint distribution is

$$\begin{pmatrix}\boldsymbol{\alpha}_{t+o} \\ \boldsymbol{\xi}_t\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}\boldsymbol{m}_{t+o} \\ \mathbf{R}^+\boldsymbol{m}_t\end{pmatrix}, \begin{pmatrix}\mathbf{P}_{t+o} & \mathrm{Cov}\left(\boldsymbol{\alpha}_{t+o}, \boldsymbol{\xi}_t\right) \\ \mathrm{Cov}\left(\boldsymbol{\xi}_t, \boldsymbol{\alpha}_{t+o}\right) & \mathbf{R}^+\mathbf{P}_t\mathbf{R}^{+\top}\end{pmatrix}\right)$$

We are now able to compute

$$\begin{aligned}
\mathrm{E}\left(\boldsymbol{\alpha}_{t+o}|\boldsymbol{\xi}_t\right) &= \boldsymbol{\mu}_{\alpha_{t+o}|\boldsymbol{\xi}_t} = \boldsymbol{m}_{t+o} + \mathrm{Cov}\left(\boldsymbol{\alpha}_{t+o}, \boldsymbol{\xi}_t\right)\mathbf{R}^\top\mathbf{P}_t^{-1}\mathbf{R}\left(\boldsymbol{\xi}_t - \mathbf{R}^+\boldsymbol{m}_t\right) \\
&= \boldsymbol{v}_t + \mathbf{V}_t\boldsymbol{\xi}_t \\
\mathbf{V}_t &= \mathrm{Cov}\left(\boldsymbol{\alpha}_{t+o}, \boldsymbol{\xi}_t\right)\mathbf{R}^\top\mathbf{P}_t^{-1}\mathbf{R} \\
\boldsymbol{v}_t &= \boldsymbol{m}_{t+o} - \mathbf{V}_t\mathbf{R}^+\boldsymbol{m}_t \\
\mathrm{Var}\left(\boldsymbol{\alpha}_{t+o}|\boldsymbol{\xi}_t\right) &= \boldsymbol{\Sigma}_{\boldsymbol{\alpha}_{t+o}|\boldsymbol{\xi}_t} = \mathbf{P}_{t+o} - \mathrm{Cov}\left(\boldsymbol{\alpha}_{t+o}, \boldsymbol{\xi}_t\right)\mathbf{R}^\top\mathbf{P}_t^{-1}\mathbf{R}\mathrm{Cov}\left(\boldsymbol{\xi}_t, \boldsymbol{\alpha}_{t+o}\right)
\end{aligned}$$

Having found this conditional distribution then we can apply similar arguments as we did before and find the following mean and covariance matrix in the proposal distribution

$$\begin{aligned}
\overleftarrow{\boldsymbol{\mu}}(\widetilde{\boldsymbol{\alpha}}_{t+o}^{(k)}, \boldsymbol{z}) &= \overleftarrow{\boldsymbol{\Sigma}}_t(\boldsymbol{z})\left(\mathbf{R}^\top\mathbf{P}_t^{-1}\boldsymbol{m}_t + \mathbf{V}_t^\top\boldsymbol{\Sigma}_{\alpha_{t+o}|\boldsymbol{\xi}_t}^{-1}(\widetilde{\boldsymbol{\alpha}}_{t+o}^{(k)} - \boldsymbol{v}_t) + \mathbf{X}_t^\top\mathbf{G}_t(\boldsymbol{z})\boldsymbol{u}_t(\boldsymbol{z})\right) \\
\overleftarrow{\boldsymbol{\Sigma}}_t(\boldsymbol{z}) &= \left(\mathbf{R}^\top\mathbf{P}_t^{-1}\mathbf{R} + \mathbf{V}_t^\top\boldsymbol{\Sigma}_{\boldsymbol{\alpha}_{t+o}|\boldsymbol{\xi}_t}^{-1}\mathbf{V}_t + \mathbf{X}_t^\top\mathbf{G}_t(\boldsymbol{z})\mathbf{X}_t\right)^{-1}
\end{aligned}$$

which replaces the mean and covariance matrix in Equation (21) and (22), $\boldsymbol{z}_t \in \mathbb{R}^r$ is the value of $\boldsymbol{\xi}_t$ at which we make the Taylor expansion, and $\mathbf{G}_t$ and $\boldsymbol{u}_t$ are defined in terms of $\boldsymbol{\xi}_t$.

3.2 Combining/Smoothing

Similarly, in Algorithm 1 we sample pairs of particles $(\boldsymbol{\alpha}_{t-1}^{(j_i)}, \widetilde{\boldsymbol{\alpha}}_{t+o}^{(k_i)})$, and sample $\boldsymbol{\xi}_t$ given each pair. We can replace the mean and covariance matrix in the proposal distribution in Equation (13) and (12) with the approximation

$$\begin{aligned}
\overleftrightarrow{\boldsymbol{\Sigma}}_t(\boldsymbol{z}) &= \left(\mathbf{Q}^{-1} + \mathbf{V}_t^\top \boldsymbol{\Sigma}_{\alpha_{t+o}|\boldsymbol{\xi}_t}^{-1} \mathbf{V}_t + \mathbf{X}_t^\top \mathbf{G}_t(\boldsymbol{z})\mathbf{X}_t\right)^{-1} \\
\overleftrightarrow{\boldsymbol{\mu}}_t(\boldsymbol{\alpha}_{t-1}^{(j_i)}, \widetilde{\boldsymbol{\alpha}}_{t+o}^{(k_i)}, \boldsymbol{z}) &= \\
&\boldsymbol{\Sigma}_t(\boldsymbol{z})\left(\mathbf{Q}^{-1}\mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j_i)} + \mathbf{V}_t^\top \boldsymbol{\Sigma}_{\alpha_{t+o}|\boldsymbol{\xi}_t}^{-1}(\widetilde{\boldsymbol{\alpha}}_{t+o}^{(k_i)} - \boldsymbol{v}_t) + \mathbf{X}_t^\top \mathbf{G}_t(\boldsymbol{z})\mathbf{X}_t\right)
\end{aligned}$$

where $\boldsymbol{z}$ is the value of $\boldsymbol{\xi}_t$ that we make the Taylor expansion at.

4 Log-Likelihood Evaluation and Parameter Estimation

In this section, I show an example of parameter estimation in the first order random walk using a Monte Carlo EM-algorithm. Then I cover the general vector autoregression model and how one can estimate the fixed effects. See Kantas et al. (2015); Del Moral et al. (2010); Schön et al. (2011) for a general coverage of parameter estimation with particle filters. Firstly though, I will remark that we can approximate the log-likelihood for a particular value of $\boldsymbol{\theta} = \{\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{\mu}_0, \mathbf{F}\}$ as described in Doucet and Johansen (2009, p. 5) and Malik and Pitt (2011, p. 193) using the forward particle filter shown in Algorithm 2. Details are omitted here for the sake of brevity.

The formulas for parameter estimation for the first order random are particularly simple. We need to estimate $\mathbf{Q}$ and $\boldsymbol{a}_0$ elements of $\boldsymbol{\varphi} = \{\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{\mu}_0\}$. We do this by running Algorithm 1 for the current $\boldsymbol{\varphi}$. This yields the following quantities from the E-step

$$\begin{aligned}
\boldsymbol{t}_t^{(\boldsymbol{\varphi})} &\approx \sum_{i=1}^{N_s} \widehat{\boldsymbol{\alpha}}_t^{(i)} \widehat{w}_t^{(i)} \\
\boldsymbol{T}_t^{(\boldsymbol{\varphi})} &\approx \sum_{i=1}^{N_s} \left(\widehat{\boldsymbol{\alpha}}_t^{(i)} - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j_{it})}\right)\left(\widehat{\boldsymbol{\alpha}}_t^{(i)} - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j_{it})}\right)^\top \widehat{w}_t^{(i)}
\end{aligned} \tag{23}$$

where we have extended the notation in Algorithm 1 such that superscript j_{it} is the index from forward cloud at time $t-1$ matching with ith smoothed particle at time t. Then we carry out the M-step by updating $\boldsymbol{\mu}_0$ and $\mathbf{Q}$ given the summary statistics above

$$\boldsymbol{\mu}_0 = \boldsymbol{t}_1^{(\boldsymbol{\varphi})} \qquad \mathbf{Q} = \frac{1}{d-1}\sum_{t=2}^{d} \mathbf{R}^+ \boldsymbol{T}_t^{(\boldsymbol{\varphi})} \mathbf{R}^{+\top} \tag{24}$$

We then take another iteration of the EM-algorithm with the new $\boldsymbol{\mu}_0$ and $\mathbf{Q}$ and repeat till a convergence criteria is satisfied.

4.1 Vector Autoregression Models

We start by defining the following matrices to cover estimation in general vector autoregression models for the latent space variable

$$\begin{aligned}
\mathbf{N} &= \left(\widehat{\boldsymbol{\alpha}}_2^{(1)}, \dots, \widehat{\boldsymbol{\alpha}}_2^{(N_s)}, \widehat{\boldsymbol{\alpha}}_3^{(1)}, \dots, \widehat{\boldsymbol{\alpha}}_3^{(N_s)}, \widehat{\boldsymbol{\alpha}}_4^{(1)}, \dots, \widehat{\boldsymbol{\alpha}}_d^{(N_s)}\right)^\top \mathbf{R}^{+\top} \\
\mathbf{M} &= \left(\boldsymbol{\alpha}_1^{(j_{12})}, \dots, \boldsymbol{\alpha}_1^{(j_{N_s 2})}, \boldsymbol{\alpha}_2^{(j_{13})}, \dots, \boldsymbol{\alpha}_2^{(j_{N_s 3})}, \boldsymbol{\alpha}_3^{(j_{14})}, \dots, \boldsymbol{\alpha}_{d-1}^{(j_{N_s d})}\right)^\top \\
\mathbf{W} &= \operatorname{diag}\left(\widehat{w}_2^{(1)}, \dots, \widehat{w}_2^{(N_s)}, \widehat{w}_3^{(1)}, \dots, \widehat{w}_3^{(N_s)}, \widehat{w}_4^{(1)}, \dots, \widehat{w}_d^{(N_s)}\right)
\end{aligned}$$

where $\operatorname{diag}(\cdot)$ is a diagonal matrix. We suppress the dependence above on the result of the E-step in a given iteration of the EM-algorithm to ease the notation. The goal is to estimate $\mathbf{F}$ and $\mathbf{Q}$ in Equation (1). We can find that the M-step maximizers are

$$\widehat{\mathbf{F}}^\top \mathbf{R}^{+\top} = \left(\mathbf{M}^\top \mathbf{W} \mathbf{M}\right)^{-1} \mathbf{M}^\top \mathbf{W} \mathbf{N} \tag{25}$$

$$\widehat{\mathbf{Q}} = \frac{1}{d-1}\left(\mathbf{N} - \mathbf{R}^+ \widehat{\mathbf{F}} \mathbf{M}\right)^\top \mathbf{W} \left(\mathbf{N} - \mathbf{R}^+ \widehat{\mathbf{F}} \mathbf{M}\right) \tag{26}$$

which are the typical vector autoregression estimators with weights. Equation (25) and (26) can easily be computed in parallel using QR decompositions as in the `bam` function in the `mgcv` package with a low memory footprint (see Wood et al., 2014). This is currently implemented. Though, the gains from a parallel implementation may be small as the computational complexity is independent of the number of observations. Consequently, the computation involved here is often fast relative to other part of the Monte Carlo EM-algorithm as the dimension of the state vector is small.

4.2 Restricted Vector Autoregression Models

Suppose that we want to restrict some of the parameters of $\mathbf{F}$ and $\mathbf{Q}$. Let

$$\begin{aligned}
(s_1, s_2, \dots, s_r)^\top &= \mathbf{J}\boldsymbol{\psi} \\
(o_{21}, o_{31}, \dots, o_{r1}, o_{32}, \dots, o_{r2}, o_{43}, \dots, o_{r,r-1})^\top &= \mathbf{K}\boldsymbol{\phi}
\end{aligned}$$

Then we can restrict the model such that

$$\sigma_i = \exp(s_i) \qquad \rho_{ij} = \frac{2}{1+\exp(-o_{ij})} - 1$$

$$\operatorname{vec}\left(\mathbf{R}^+\mathbf{F}\right) = \mathbf{G}\boldsymbol{\theta} \qquad \mathbf{Q} = \mathbf{V}\mathbf{C}\mathbf{V}$$

$$\mathbf{V} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \sigma_r \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} 1 & \rho_{21} & \cdots & \rho_{r1} \\ \rho_{21} & 1 & \ddots & \rho_{r2} \\ \vdots & \ddots & \ddots & \vdots \\ \rho_{r1} & \dots & \rho_{r,r-1} & 1 \end{pmatrix}$$

and where $\text{vec}(\cdot)$ is the vectorization function which stacks the the columns of a matrix from left to right. E.g.,

$$\begin{aligned}\text{vec}(\mathbf{A}) &= (a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{32}, a_{13}, a_{23}, a_{33})^\top \\ \mathbf{A} &= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\end{aligned}$$

$\mathbf{G} \in \mathbb{R}^{rp \times g}$ is a known matrix with $g \leq rp$ and we assume that it has full column rank. Similarly, $\mathbf{J} \in \mathbb{R}^{r \times l}$ with $l \leq r$ and $\mathbf{K} \in \mathbb{R}^{r(r-1)/2 \times k}$ with $k \leq r(r-1)/2$. Both are known and have full column rank. We assume that $\mathbf{G}$ is such that $\mathbf{F}$ is non-singular for some $\boldsymbol{\theta}$ since Equation (9) used. Further, we assume that $\mathbf{J}$ and $\mathbf{K}$ are such that $\mathbf{Q}$ is a positive definite matrix for some $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ pair. $\mathbf{V}$ is a diagonal matrix containing the standard deviations and $\mathbf{C}$ is the correlation matrix.

We cannot jointly maximize $\boldsymbol{\theta}$, $\boldsymbol{\psi}$, and $\boldsymbol{\phi}$ analytically but we can maximize $\boldsymbol{\theta}$ analytically conditional on $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$. Hence, we can employ a Monte Carlo expectation conditional maximization algorithm in which we take two so-called conditional maximization steps (see Meng and Rubin, 1993, for the, non-Monte Carlo, expectation maximization algorithm). The first conditional maximization step is

$$\boldsymbol{\theta}^{(i+1)} = \mathbf{G}^+ \left(\mathbf{Q}^{(i)} \otimes \left(\mathbf{M}^\top \mathbf{W} \mathbf{M}\right)^{-1}\right) \mathbf{G}^{+\top} \mathbf{G}^\top \text{vec}\left(\mathbf{M}^\top \mathbf{W} \mathbf{N} \mathbf{Q}^{-(i)}\right) \tag{27}$$

where $\otimes$ is the Kronecker product and $\mathbf{Q}^{-(i)}$ is the inverse of $\mathbf{Q}^{(i)}$. Equation (27) is easily computed with the QR decomposition we make to compute for Equation (25). Having obtained the new $\boldsymbol{\theta}^{(i+1)}$, we update $\mathbf{F}$ and denote the new estimate $\widehat{\mathbf{F}}^{(i+1)}$. The second conditional maximization step which updates $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ is

$$\begin{aligned}\mathbf{Z} &= \left(\mathbf{N} - \mathbf{R}^+ \widehat{\mathbf{F}}^{(i+1)} \mathbf{M}\right)^\top \mathbf{W} \left(\mathbf{N} - \mathbf{R}^+ \widehat{\mathbf{F}}^{(i+1)} \mathbf{M}\right) \\ \boldsymbol{\psi}^{(i+1)}, \boldsymbol{\phi}^{(i+1)} &= \underset{\boldsymbol{\psi}, \boldsymbol{\phi}}{\arg\max} -(d-1) \log |\mathbf{Q}(\boldsymbol{\psi}, \boldsymbol{\phi})| - \text{tr}\left(\mathbf{Q}(\boldsymbol{\psi}, \boldsymbol{\phi})^{-1} \mathbf{Z}\right)\end{aligned}$$

which can be done numerically. We have made $\mathbf{Q}$'s dependence on $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ explicit to emphasize which factors are affected. $\mathbf{C}$ may not be a valid correlation matrix for all $\boldsymbol{\phi} \in \mathbb{R}^k$ for some choices of $\mathbf{K}$. Thus, the numerical optimization algorithm is constrained to valid correlation matrices. This completes the two conditional maximization steps. The next E-step is then performed using $\boldsymbol{\theta}^{(i+1)}$, $\boldsymbol{\psi}^{(i+1)}$, $\boldsymbol{\phi}^{(i+1)}$. Meng and Rubin (1993, see the discussion) comments that it may be beneficial to perform an E-step between each conditional maximization step when the E-step is relatively cheap. This is not the case here since all the above computations are independent of the number observations, $n_{\max}$. Thus, if we have a moderately large number of observations at each time point relative to the dimension of the state vector, then we will use most of the computation time performing the E-step.

4.3 Estimating Fixed Effect Coefficients

Next, we turn to estimating the fixed effect coefficients, $\boldsymbol{\omega}$, in Equation (1). If we assume that observations, y_{it}s, are from an exponential family then it is easy to show that the M-step estimator amounts to generalized linear model with N_s observations for each y_{it}

which differ only by an offset term and a weight. The offset term comes from the $\boldsymbol{x}_{it}^\top \mathbf{R}^+ \widehat{\boldsymbol{\alpha}}_j^{(t)}$ term in Equation (1) for each of the $j = 1, \dots, N_s$ smoothed particles. The corresponding weights are the smoothed weights, $\widehat{w}_j^{(t)}$. The problem can be solved in parallel using QR decompositions as in Section 4.1. This is what is done in the current implementation.

Currently, I only take one iteration of the iteratively re-weighted least squares. I gather I have to repeat till convergence though... This is however not nice computationally and the difference in the estimate from one M-step iteration to the next is very minor when you only take one iteratively re-weighted least square iteration...

5 Other Filter and Smoother Options

The $\mathcal{O}\left(N^2\right)$ two-filter smoother in Fearnhead et al. (2010) is going to be computationally expensive as an approximation is going to be needed for Equation (8) in their article. The non-auxiliary version in Briers et al. (2009) is more feasible as it only requires evaluation of f in the smoothing part of the generalized two-filter smoother (see Equation (46) in their paper). Similar conclusions applies to the forward smoother in Del Moral et al. (2010) and the backward smoother as presented in Kantas et al. (2015). Both have a $\mathcal{O}\left(N^2\right)$ computational cost.

Despite the $\mathcal{O}\left(N^2\right)$ cost of the method in Briers et al. (2009) and Del Moral et al. (2010) they may still be useful as the computational cost in the smoothing step is independent of the number of observations, n_{max}. Further, the computational cost can be reduced to $\mathcal{O}\left(N \log(N)\right)$ run times with approximations like in Klaas et al. (2006).

The method in Malik and Pitt (2011, see particularly section 6.2 on page 203) can be used to do continuous likelihood approximations as a function of the unknown parameters. Though, I am not sure how well this method scale with higher state dimensions, p.

Kantas et al. (2015) show empirically that it may be worth just using a forward filter. However, the example is with an univariate outcome ($n_{\text{max}} = 1$ not to be confused with the number of periods d). In the problems shown in this vignette, the computational complexity of the forward filter is at least $\mathcal{O}\left(dNn_{\text{max}}p\right)$. Every new particle yields an $\mathcal{O}\left(dn_{\text{max}}p\right)$ cost which is expensive due to the large number of outcomes, n_{max}. Thus, the considerations are different and a $\mathcal{O}\left(dNn_{\text{max}}p + N^2\right)$ method will not make a big difference unless N is large.

6 Generalized Two-Filter Smoother

The $\mathcal{O}\left(N^2\right)$ smother from Briers et al. (2009) is also implemented as it is feasible for a moderate number of particles (though, we can use the approximations in Kantas et al., 2015 to reduce the computational complexity). It is shown in Algorithm 4. The weights in Equation (30) comes from the generalized two-filter formula. To motivate the smoother, we use

$$p\left(\boldsymbol{y}_{t:d} \vert \boldsymbol{\alpha}_t\right) = \widetilde{p}\left(\boldsymbol{y}_{t:d}\right) \frac{\widetilde{p}\left(\boldsymbol{\alpha}_t \vert \boldsymbol{y}_{t:d}\right)}{\gamma_t(\boldsymbol{\alpha}_t)}$$

to generalize the two-filter formula in Kitagawa (1994) as follows

$$
\begin{aligned}
p\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{1:d}\right) &= \frac{p\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{1:t-1}\right) p\left(\boldsymbol{y}_{t:d} | \boldsymbol{\alpha}_t\right)}{p\left(\boldsymbol{y}_{t:d} | \boldsymbol{y}_{1:t-1}\right)} \\
&\propto p\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{1:t-1}\right) p\left(\boldsymbol{y}_{t:d} | \boldsymbol{\alpha}_t\right) \\
&= p\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{1:t-1}\right) \widetilde{p}\left(\boldsymbol{y}_{t:d}\right) \frac{\widetilde{p}\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{t:d}\right)}{\gamma_t(\boldsymbol{\alpha}_t)} \\
&\propto p\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{1:t-1}\right) \frac{\widetilde{p}\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{t:d}\right)}{\gamma_t(\boldsymbol{\alpha}_t)} \\
&= \widetilde{p}\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{t:d}\right) \frac{\left[\int p\left(\boldsymbol{\alpha}_{t-1} | \boldsymbol{y}_{1:t-1}\right) f\left(\boldsymbol{\alpha}_t | \boldsymbol{\alpha}_{t-1}\right) \mathrm{d}\boldsymbol{\alpha}_{t-1}\right]}{\gamma_t(\boldsymbol{\alpha}_t)} \\
&\mathrel{\underset{\sim}{\propto}} \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{\widetilde{\boldsymbol{\alpha}}_t^{(i)}}\left(\boldsymbol{\alpha}_t\right) \frac{\left[\sum_{j=1}^{N} w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]}{\gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)}
\end{aligned} \tag{28}
$$

where $\mathrel{\underset{\sim}{\propto}}$ means approximately proportional. Similar arguments leads to

$$
\begin{aligned}
p\left(\boldsymbol{\alpha}_{t-1:t} | \boldsymbol{y}_{1:d}\right) &\propto p\left(\boldsymbol{\alpha}_{t-1:t} | \boldsymbol{y}_{1:t-1}\right) p\left(\boldsymbol{y}_{t:d} | \boldsymbol{\alpha}_{t-1:t}\right) \\
&= f\left(\boldsymbol{\alpha}_t | \boldsymbol{\alpha}_{t-1}\right) p\left(\boldsymbol{\alpha}_{t-1} | \boldsymbol{y}_{1:t-1}\right) p\left(\boldsymbol{y}_{t:d} | \boldsymbol{\alpha}_t\right) \\
&\propto f\left(\boldsymbol{\alpha}_t | \boldsymbol{\alpha}_{t-1}\right) p\left(\boldsymbol{\alpha}_{t-1} | \boldsymbol{y}_{1:t-1}\right) \frac{\widetilde{p}\left(\boldsymbol{\alpha}_t | \boldsymbol{y}_{t:d}\right)}{\gamma_t(\boldsymbol{\alpha}_t)} \\
&\mathrel{\underset{\sim}{\propto}} \sum_{i=1}^{N} \sum_{j=1}^{N} \widetilde{w}_t^{(i)} \delta_{\widetilde{\boldsymbol{\alpha}}_t^{(i)}}\left(\boldsymbol{\alpha}_t\right) \frac{\left[\sum_{k=1}^{N} w_{t-1}^{(k)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(k)}\right)\right]}{\gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)} \\
&\qquad \cdot \frac{w_{t-1}^{(j)} \delta_{\boldsymbol{\alpha}_{t-1}^{(j)}}\left(\boldsymbol{\alpha}_{t-1}\right) f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)}{\left[\sum_{k=1}^{N} w_{t-1}^{(k)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(k)}\right)\right]} \\
&= \sum_{i=1}^{N} \sum_{j=1}^{N} \widehat{w}_t^{(i,j)} \delta_{\widetilde{\boldsymbol{\alpha}}_t^{(i)}}\left(\boldsymbol{\alpha}_t\right) \delta_{\boldsymbol{\alpha}_{t-1}^{(j)}}\left(\boldsymbol{\alpha}_{t-1}\right)
\end{aligned}
$$

where

$$
\widehat{w}_t^{(i,j)} = \widehat{w}_t^{(i)} \frac{w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)}{\left[\sum_{j=1}^{N} w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]} \tag{29}
$$

We need the latter for the Monte Carlo EM-algorithm.

Algorithm 4 $\mathcal{O}\left(N^2\right)$ generalized two-filter smoother using the method suggested by Briers et al. (2009).

Input:

$\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{a}_0, \mathbf{X}_1, \ldots, \mathbf{X}_d, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_d, R_1, \ldots, R_d, \boldsymbol{\omega}$

procedure FILTER FORWARD

Run a forward particle filter to get particle clouds $\left\{\boldsymbol{\alpha}_t^{(j)}, w_t^{(j)}, \beta_{t+1}^{(j)}\right\}_{j=1,\ldots,N}$ approximating $p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:t}\right)$ for $t=0,1,\ldots,d$. See Algorithm 2.

procedure FILTER BACKWARDS

Run a similar backward filter to get $\left\{\widetilde{\boldsymbol{\alpha}}_t^{(k)}, \widetilde{w}_t^{(k)}, \widetilde{\beta}_{t-1}^{(k)}\right\}_{k=1,\ldots,N}$ approximating $\widetilde{p}\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{t:d}\right)$ for $t=d+1,d,d-1,\ldots,1$. See Algorithm 3.

procedure SMOOTH (COMBINE)

for $t=1,\ldots,d$ **do**

Assign each backward filter particle a smoothing weight given by

$$\widehat{w}_t^{(i)} \propto \widetilde{w}_t^{(i)} \frac{\left[\sum_{j=1}^N w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\middle|\boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]}{\gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)} \tag{30}$$

With the result above, we can show the arguments behind the smoother from Fearnhead et al. (2010). Similar to Equation (28), we find that

$$\begin{aligned}
p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:d}\right) &\propto p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:t-1}\right) p\left(\boldsymbol{y}_{t:d}|\boldsymbol{\alpha}_t\right) \\
&= p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:t-1}\right) g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) p\left(\boldsymbol{y}_{t+1:d}|\boldsymbol{\alpha}_t\right) \\
&= \int f\left(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}\right) p\left(\boldsymbol{\alpha}_{t-1}|\boldsymbol{y}_{1:t-1}\right) \mathrm{d}\boldsymbol{\alpha}_{t-1} g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) \\
&\qquad \cdot \int f\left(\boldsymbol{\alpha}_{t+1}|\boldsymbol{\alpha}_t\right) p\left(\boldsymbol{y}_{t+1:d}|\boldsymbol{\alpha}_{t+1}\right) \mathrm{d}\boldsymbol{\alpha}_{t+1} \\
&\propto \int f\left(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}\right) p\left(\boldsymbol{\alpha}_{t-1}|\boldsymbol{y}_{1:t-1}\right) \mathrm{d}\boldsymbol{\alpha}_{t-1} g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) \\
&\qquad \cdot \int f\left(\boldsymbol{\alpha}_{t+1}|\boldsymbol{\alpha}_t\right) \frac{\widetilde{p}\left(\boldsymbol{\alpha}_{t+1}|\boldsymbol{y}_{t+1:d}\right)}{\gamma_{t+1}(\boldsymbol{\alpha}_{t+1})} \mathrm{d}\boldsymbol{\alpha}_{t+1} \\
&\underset{\sim}{\propto} \sum_{j=1}^N \sum_{k=1}^N f\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)} g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\middle|\boldsymbol{\alpha}_t\right) \frac{\widetilde{w}_{t+1}^{(k)}}{\gamma_{t+1}(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)})}
\end{aligned}$$

Thus, we can sample $\boldsymbol{\alpha}_t$ from a proposal distribution given the time $t-1$ forward filter particle, $\boldsymbol{\alpha}_{t-1}^{(j)}$, and time $t+1$ backward filter particle, $\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}$, for all N^2 particle pairs. Alternatively, we can sample the $t-1$ and $t+1$ particles independently which yields Algorithm 1.

Further, we can find that

$$\begin{aligned}
p\left(\boldsymbol{\alpha}_{t-1:t}|\boldsymbol{y}_{1:d}\right) &= p\left(\boldsymbol{\alpha}_{t-1:t}|\boldsymbol{y}_{1:t-1}\right) g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_{t-1:t}\right) p\left(\boldsymbol{y}_{t+1:d}|\boldsymbol{\alpha}_{t-1:t}\right) \\
&\propto f\left(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}\right) p\left(\boldsymbol{\alpha}_{t-1}|\boldsymbol{y}_{1:t-1}\right) g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) \\
&\qquad\qquad \cdot \int f\left(\boldsymbol{\alpha}_{t+1}|\boldsymbol{\alpha}_t\right) \frac{\widetilde{p}\left(\boldsymbol{\alpha}_{t+1}|\boldsymbol{y}_{t+1:d}\right)}{\gamma_{t+1}(\boldsymbol{\alpha}_{t+1})} \mathrm{d}\boldsymbol{\alpha}_{t+1} \\
&\overset{\propto}{\sim} \sum_{i=1}^{N_s} \delta_{\widehat{\boldsymbol{\alpha}}_t^{(i)}}\left(\boldsymbol{\alpha}_t\right) \delta_{\boldsymbol{\alpha}_{t-1}^{(j_i)}}\left(\boldsymbol{\alpha}_t\right) f\left(\widehat{\boldsymbol{\alpha}}_t^{(i)}\middle|\boldsymbol{\alpha}_{t-1}^{(j_i)}\right) w_{t-1}^{(j_i)} g_t\left(\boldsymbol{y}_t\middle|\widehat{\boldsymbol{\alpha}}_t^{(i)}\right) \\
&\qquad\qquad \cdot \int f\left(\boldsymbol{\alpha}_{t+1}\middle|\widehat{\boldsymbol{\alpha}}_t^{(i)}\right) \frac{\widetilde{p}\left(\boldsymbol{\alpha}_{t+1}|\boldsymbol{y}_{t+1:d}\right)}{\gamma_{t+1}(\boldsymbol{\alpha}_{t+1})} \mathrm{d}\boldsymbol{\alpha}_{t+1} \\
&\overset{\propto}{\sim} \sum_{i=1}^{N_s} \widehat{w}_t^{(i)} \delta_{\widehat{\boldsymbol{\alpha}}_t^{(i)}}\left(\boldsymbol{\alpha}_t\right) \delta_{\boldsymbol{\alpha}_{t-1}^{(j_i)}}\left(\boldsymbol{\alpha}_t\right)
\end{aligned}$$

where superscripts j_i are used as in Algorithm 1 which implicitly dependent on t.

7 Gradient and Observed Information Matrix

An alternative to the Monte Carlo EM-algorithm is to approximate the gradient and use it to perform the maximization with a gradient descent algorithm. Moreover, one may be interested in the observed information matrix. Two methods are implemented in order to make such approximations. The first method is the method covered in Cappé et al. (2005, section 8.3 and chapter 11). It has the advantage that it uses the output from the forward particle filter. However, the variance of the estimates increase at least quadratically in time, d. An alternative is to use the method shown by Poyiadjis et al. (2011). Like the smoothing algorithm from Briers et al. (2009), this method has the disadvantage of having a computational complexity which is quadratic in the number of particles, N.

I will give a brief introduction to the two methods in this section. What is presented here closely follow Poyiadjis et al. (2011). First, we will need some notation. We denote the complete data log-likelihood by

$$\begin{aligned}
c\left(\boldsymbol{y}_{1:t}, \boldsymbol{\alpha}_{0:t}\right) &= \log h\left(\boldsymbol{y}_{1:t}, \boldsymbol{\alpha}_{0:t}\right) \\
h\left(\boldsymbol{y}_{1:t}, \boldsymbol{\alpha}_{0:t}\right) &= \nu\left(\boldsymbol{\alpha}_0\right) \prod_{k=1}^{t} g_k\left(\boldsymbol{y}_k|\boldsymbol{\alpha}_k\right) f\left(\boldsymbol{\alpha}_k|\boldsymbol{\alpha}_{k-1}\right)
\end{aligned}$$

where ν is the density function of the state vector at time zero, all functions may implicitly depend on the unknown parameters, and the dimension of the arguments to c and h is given by the superscript of the arguments. A direct application of the results by Louis (1982) shows that the gradient of the observed data log-likelihood

$$o\left(\boldsymbol{y}_{1:t}\right) = \log \int h\left(\boldsymbol{y}_{1:t}, \boldsymbol{a}_{0:t}\right) \mathrm{d}\boldsymbol{a}_{0:t}$$

w.r.t. the unknown parameters are

$$\begin{aligned}
\nabla o\left(\boldsymbol{y}_{1:t}\right) &= \frac{\partial}{\partial \boldsymbol{\theta}} \log \int h\left(\boldsymbol{y}_{1:t}, \boldsymbol{a}_{0:t}\right) \mathrm{d}\boldsymbol{a}_{0:t} = \frac{\int h'\left(\boldsymbol{y}_{1:t}, \boldsymbol{a}_{0:t}\right) d\boldsymbol{a}_{0:t}}{\int h\left(\boldsymbol{y}_{1:t}, \boldsymbol{a}_{0:t}\right) \mathrm{d}\boldsymbol{a}_{0:t}} \qquad (31) \\
&= \int c'\left(\boldsymbol{y}_{1:t}, \boldsymbol{a}_{0:t}\right) p\left(\boldsymbol{a}_{0:t}|\boldsymbol{y}_{1:t}\right) \mathrm{d}\boldsymbol{a}_{0:t}
\end{aligned}$$

where $\boldsymbol{\theta}$ are the unknown parameters in the model, derivatives are w.r.t. $\boldsymbol{\theta}$, and $p\left(\boldsymbol{a}_{0:t}|\boldsymbol{y}_{1:t}\right)$ is the conditional density function of $\boldsymbol{a}_{0:t}$ given $\boldsymbol{y}_{1:t}$. Moreover, the Hessian is

$$\begin{aligned}\nabla^2 o\left(\boldsymbol{y}_{1:t}\right) &= \frac{\int h''\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_{0:t}\right)d\boldsymbol{a}_{0:t}}{\int h\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_{0:t}\right)\mathrm{d}\boldsymbol{a}_{0:t}} - \nabla o\left(\boldsymbol{y}_{1:t}\right)\nabla o\left(\boldsymbol{y}_{1:t}\right)^\top \\ &= \int c''\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_{0:t}\right)p\left(\boldsymbol{a}_{0:t}|\boldsymbol{y}_{1:t}\right)\mathrm{d}\boldsymbol{a}_{0:t} \qquad (32) \\ &\quad + \int c'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_{0:t}\right)c'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_{0:t}\right)^\top p\left(\boldsymbol{a}_{0:t}|\boldsymbol{y}_{1:t}\right)\mathrm{d}\boldsymbol{a}_{0:t} - \nabla o\left(\boldsymbol{y}_{1:t}\right)\nabla o\left(\boldsymbol{y}_{1:t}\right)^\top\end{aligned}$$

We can use that the forward particle filter yields not just an approximation of $p\left(\boldsymbol{a}_d|\boldsymbol{y}_{1:d}\right)$ but the entire path $p\left(\boldsymbol{a}_{0:d}|\boldsymbol{y}_{1:d}\right)$. That is, we can use the weights at time d from Equation (17) and make a discrete approximation of Equation (31) and (32) as shown in Cappé et al. (2005). However, the variance of the estimates grows at-least quadratically in d as shown by Poyiadjis et al. (2011). The issue is that for larger d, then few if not only one unique value of the initial state vector values ($\boldsymbol{\alpha}_i$ with $0 \leq i \ll d$) are present in the discrete approximation.

As an alternative, Poyiadjis et al. (2011) develop a marginal version of Equation (31) and (32). That is,

$$\begin{aligned}\tilde{c}\left(\boldsymbol{y}_{1:t},\boldsymbol{\alpha}_t\right) &= \log\tilde{h}\left(\boldsymbol{y}_{1:t},\boldsymbol{\alpha}_t\right) \\ \tilde{h}\left(\boldsymbol{y}_{1:t},\boldsymbol{\alpha}_t\right) &= \begin{cases} g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right)\int f\left(\boldsymbol{\alpha}_t|\boldsymbol{a}_{t-1}\right)\tilde{h}\left(\boldsymbol{y}_{1:(t-1)},\boldsymbol{a}_{t-1}\right)\mathrm{d}\boldsymbol{a}_{t-1} & t>0 \\ \nu\left(\boldsymbol{a}_t\right) & t=0\end{cases} \qquad (33) \\ \nabla o\left(\boldsymbol{y}_{1:t}\right) &= \int \tilde{c}'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)p\left(\boldsymbol{a}_t|\boldsymbol{y}_{1:t}\right)\mathrm{d}\boldsymbol{a}_t \qquad (34) \\ \nabla^2 o\left(\boldsymbol{y}_{1:t}\right) &= \int \tilde{c}''\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)p\left(\boldsymbol{a}_t|\boldsymbol{y}_{1:t}\right)\mathrm{d}\boldsymbol{a}_t \\ &\quad + \int \tilde{c}'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)\tilde{c}'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)^\top p\left(\boldsymbol{a}_t|\boldsymbol{y}_{1:t}\right)\mathrm{d}\boldsymbol{a}_t - \nabla o\left(\boldsymbol{y}_{1:t}\right)\nabla o\left(\boldsymbol{y}_{1:t}\right)^\top \\ &= \int\left(\frac{\tilde{h}''\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)}{\tilde{h}\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)} - \tilde{c}'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)\tilde{c}'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)^\top\right)p\left(\boldsymbol{a}_t|\boldsymbol{y}_{1:t}\right)\mathrm{d}\boldsymbol{a}_t \qquad (35) \\ &\quad + \int \tilde{c}'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)\tilde{c}'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)^\top p\left(\boldsymbol{a}_t|\boldsymbol{y}_{1:t}\right)\mathrm{d}\boldsymbol{a}_t - \nabla o\left(\boldsymbol{y}_{1:t}\right)\nabla o\left(\boldsymbol{y}_{1:t}\right)^\top\end{aligned}$$

While there is no analytical expression for the derivatives then one can establish a point-wise approximation recursively for $c'\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)$ and $c''\left(\boldsymbol{y}_{1:t},\boldsymbol{a}_t\right)$ as suggested by Poyiadjis et al. (2011). To see this, let

$$s_t\left(\boldsymbol{\alpha}_t,\boldsymbol{\alpha}_{t-1}\right) = \log g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) + \log f\left(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}\right)$$

Then

$$\begin{aligned}\tilde{h}'\left(\boldsymbol{y}_{1:t},\boldsymbol{\alpha}_t\right) = o\left(\boldsymbol{y}_{1:t-1}\right)g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right)\int f\left(\boldsymbol{\alpha}_t|\boldsymbol{a}_{t-1}\right)p\left(\boldsymbol{a}_{t-1}|\boldsymbol{y}_{1:t-1}\right) \qquad (36) \\ \cdot\left(s_t'\left(\boldsymbol{\alpha}_t,\boldsymbol{a}_{t-1}\right)+\tilde{c}'\left(\boldsymbol{y}_{1:(t-1)},\boldsymbol{a}_{t-1}\right)\right)\mathrm{d}\boldsymbol{a}_{t-1}\end{aligned}$$

Taking the ratio of Equation (36) and (33) yields $\tilde{c}'(\boldsymbol{y}_{1:t}, \boldsymbol{\alpha}_t)$ in Equation (34). Moreover, for Equation (35)

$$\begin{aligned}\tilde{h}''(\boldsymbol{y}_{1:t}, \boldsymbol{\alpha}_t) = o(\boldsymbol{y}_{1:t-1})\, g_t(\boldsymbol{y}_t|\boldsymbol{\alpha}_t) \int f(\boldsymbol{\alpha}_t|\boldsymbol{a}_{t-1})\, p(\boldsymbol{a}_{t-1}|\boldsymbol{y}_{1:t-1}) \\ \cdot \Bigg(\left(s_t'(\boldsymbol{\alpha}_t, \boldsymbol{a}_{t-1}) + \tilde{c}'\left(\boldsymbol{y}_{1:(t-1)}, \boldsymbol{a}_{t-1}\right)\right)\left(s_t'(\boldsymbol{\alpha}_t, \boldsymbol{a}_{t-1}) + \tilde{c}'\left(\boldsymbol{y}_{1:(t-1)}, \boldsymbol{a}_{t-1}\right)\right)^\top \\ + s_t''(\boldsymbol{\alpha}_t, \boldsymbol{a}_{t-1}) + \tilde{c}''\left(\boldsymbol{y}_{1:(t-1)}, \boldsymbol{a}_{t-1}\right) \Bigg) \mathrm{d}\boldsymbol{a}_{t-1}\end{aligned}$$

where we again take the ratio with (33). Unlike before, we need to evaluate two ratios, $\tilde{h}'(\boldsymbol{y}_{1:t}, \boldsymbol{a}_t) / \tilde{h}(\boldsymbol{y}_{1:t}, \boldsymbol{a}_t)$ and $\tilde{h}''(\boldsymbol{y}_{1:t}, \boldsymbol{a}_t) / \tilde{h}(\boldsymbol{y}_{1:t}, \boldsymbol{a}_t)$, which require evaluation of expressions of the form

$$\frac{\int f(\boldsymbol{\alpha}_t|\boldsymbol{a}_{t-1})\, p(\boldsymbol{a}_{t-1}|\boldsymbol{y}_{1:t-1})\, \kappa_t(\boldsymbol{\alpha}_t, \boldsymbol{a}_{t-1})\, \mathrm{d}\boldsymbol{a}_{t-1}}{\int f(\boldsymbol{\alpha}_t|\boldsymbol{a}_{t-1})\, p(\boldsymbol{a}_{t-1}|\boldsymbol{y}_{1:t-1})\, \mathrm{d}\boldsymbol{a}_{t-1}} \tag{37}$$

for some function κ_t. To do so, redefine the weights in Equation (17) in the forward particle filter shown in Algorithm 2 to

$$w_t^{(i)} \propto \frac{g_t\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_t^{(i)}\right) \sum_{j=1}^N f\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)}}{q\left(\boldsymbol{\alpha}_t^{(i)} \middle| \left\{\left(\boldsymbol{\alpha}_{t-1}^{(j)}, w_{t-1}^{(j)}\right)\right\}_{j=1,\dots,N}, \boldsymbol{y}_t\right)}$$

where we have made it explicit that the proposal distribution may depend on the previous particle cloud and assume that we use the same number of particles at time 0. Further, define the weights

$$\bar{w}_t^{(i,j)} = \frac{f\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)}}{\sum_{k=1}^N f\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(k)}\right) w_{t-1}^{(k)}} \tag{38}$$

Now a discrete approximation of expression in Equation (37) is given by

$$\sum_{j=1}^N \bar{w}_t^{(i,j)} \kappa_t\left(\boldsymbol{\alpha}_t^{(i)}, \boldsymbol{\alpha}_{t-1}^{(j)}\right)$$

Thus, the recursive formula for the gradient approximation is

$$\begin{aligned}\boldsymbol{\zeta}_t^{(i)} &= \sum_{j=1}^N \bar{w}_t^{(i,j)} \left(s_t'\left(\boldsymbol{\alpha}_t^{(i)}, \boldsymbol{\alpha}_{t-1}^{(j)}\right) + \boldsymbol{\zeta}_{t-1}^{(j)}\right) \qquad (39)\\ \nabla o(\boldsymbol{y}_{1:t}) &\approx \sum_{i=1}^N w_t^{(i)} \boldsymbol{\zeta}_t^{(i)}\end{aligned}$$

and for the Hessian we have

$$\begin{aligned}\boldsymbol{\Upsilon}_t^{(i)} = \sum_{j=1}^N \bar{w}_t^{(i,j)} \Bigg(\left(s_t'\left(\boldsymbol{\alpha}_t^{(i)}, \boldsymbol{\alpha}_{t-1}^{(j)}\right) + \boldsymbol{\zeta}_{t-1}^{(j)}\right)\left(s_t'\left(\boldsymbol{\alpha}_t^{(i)}, \boldsymbol{\alpha}_{t-1}^{(j)}\right) + \boldsymbol{\zeta}_{t-1}^{(j)}\right)^\top \qquad (40)\\ + s_t''\left(\boldsymbol{\alpha}_t^{(i)}, \boldsymbol{\alpha}_{t-1}^{(j)}\right) + \boldsymbol{\Upsilon}_{t-1}^{(j)} \Bigg) - \boldsymbol{\zeta}_t^{(i)} \boldsymbol{\zeta}_t^{(i)\top}\end{aligned}$$

such that

$$\nabla^2 o\left(\boldsymbol{y}_{1:t}\right) \approx \sum_{i=1}^N w_t^{(i)}\left(\boldsymbol{\zeta}_t^{(i)}\boldsymbol{\zeta}_t^{(i)\top} + \boldsymbol{\Upsilon}_t^{(i)}\right) - \nabla o\left(\boldsymbol{y}_{1:t}\right)\nabla o\left(\boldsymbol{y}_{1:t}\right)^\top$$

The issue with the latter method is that the method has an $\mathcal{O}\left(N^2\right)$ computational complexity because of the sums in Equation (38), (39), and (40). A particular type of particle filters that are well suited for approximations like those in Equation (37) are the so-called independent particle filters suggest by Lin et al. (2005). The key point in these filters is that they use a proposal distribution which only depends on the observed outcome, $\boldsymbol{y}_t$, or also the previous particle cloud but not any particular particle.

An alternative to the methods in the `dynamichazard` package is the `mssm` package. It contains both the method shown in Cappé et al. (2005) and the method suggested by Poyiadjis et al. (2011) but for more general models. Moreover, the `mssm` package has an implementation of the dual k-d tree approximation method like in Klaas et al. (2006). This reduces the average-case complexity to $\mathcal{O}\left(N\log N\right)$ and thus it allows one to use substantially more particles. Lastly, the `mssm` also allows for two types of antithetic variables like those suggest by Durbin and Koopman (1997). This decreases the variance of the estimates for the same computational cost.

References

Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61, Jun 2009. ISSN 1572-9052. doi: 10.1007/s10463-009-0236-2. URL `https://doi.org/10.1007/s10463-009-0236-2`.

Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer-Verlag New York, 2005. ISBN 978-0-387-40264-2, 978-1-4419-2319-6.

Pierre Del Moral, Arnaud Doucet, and Sumeetpal Singh. Forward smoothing using sequential monte carlo. *arXiv preprint arXiv:1012.5390*, 2010.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

Randal Douc and Olivier Cappé. Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69. IEEE, 2005.

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

J. Durbin and S. J. Koopman. Monte carlo maximum likelihood estimation for non-Gaussian state space models. *Biometrika*, 84(3):669–684, 1997. ISSN 00063444. URL `http://www.jstor.org/stable/2337587`.

Paul Fearnhead, David Wyncoll, and Jonathan Tawn. A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447–464, 2010.

Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3):328–351, 2015.

Genshiro Kitagawa. The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4):605–623, Dec 1994. ISSN 1572-9052. doi: 10.1007/BF00773470. URL `https://doi.org/10.1007/BF00773470`.

Genshiro Kitagawa. Monte carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

Mike Klaas, Mark Briers, Nando De Freitas, Arnaud Doucet, Simon Maskell, and Dustin Lang. Fast particle smoothing: If i had a million particles. In *Proceedings of the 23rd international conference on Machine learning*, pages 481–488. ACM, 2006.

Ming T. Lin, Junni L. Zhang, Qiansheng Cheng, and Rong Chen. Independent particle filters. *Journal of the American Statistical Association*, 100(472):1412–1421, 2005. ISSN 01621459. URL `http://www.jstor.org/stable/27590681`.

Thomas A. Louis. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 44(2):226–233, 1982. ISSN 00359246. URL `http://www.jstor.org/stable/2345828`.

Sheheryar Malik and Michael K Pitt. Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, 165(2):190–209, 2011.

Xiao-Li Meng and Donald B. Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993. ISSN 00063444. URL `http://www.jstor.org/stable/2337198`.

Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

George Poyiadjis, Arnaud Doucet, and Sumeetpal S. Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011. ISSN 00063444. URL `http://www.jstor.org/stable/29777165`.

Thomas B Schön, Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

Simon Wood, Yannig Goude, and Simon Shaw. Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 64(1):139–155, 2014. doi: 10.1111/rssc.12068. URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/rssc.12068`.