Particle filters in the dynamichazard package

Benjamin Christoffersen

May 7, 2018

This is vignette covers the particle filter for the `dynamichazard` package in `R`. See `https://cran.r-project.org/web/packages/dynamichazard/vignettes/ddhazard.pdf` for more information on the package.

I assume that you are familiar with the setup in the `dynamichazard` package and the other estimation methods in the package. Further, some prior knowledge of particle filters is required. If you lag knowledge of particle filter then Doucet and Johansen (2009) provides a tutorial on particle filters and Kantas et al. (2015) covers parameter estimation with particle filters. This vignette relies heavily on Fearnhead et al. (2010).

1 Method

Model

The model is:

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

where I denote the conditional densities as $\boldsymbol{y}_t \sim g\left(\cdot | \, \mathbf{R}^+ \boldsymbol{\alpha}_t\right)$ and $\boldsymbol{\alpha}_t \sim f\left(\cdot | \, \boldsymbol{\alpha}_{t-1}\right)$. We are in a survival analysis setting where the simplest model has an indicator of death of individual i in time t such that $y_{it} \in \{0, 1\}$ where η_{it} is the linear predictor where we use the logistic function as the link function. For each $t = 1, \dots, d$, we a have risk set given by $R_t \subseteq \{1, 2, \dots, n\}$. Further, We let $n_t = |R_t|$ and $n_{\max} = \max_{t=1,\dots d} = n_t$. 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 vector of time-varying coefficients. The problems we are looking at have $n_{\max} \gg p \geq r$ (e.g. $n_{\max} = 100000$ and $r = 20$).

Superscript $+$ denotes the Moore-Penrose inverse, the i'th row of $\mathbf{X}_t$ is $\boldsymbol{x}_{it}$, $\boldsymbol{x}_{it}, \boldsymbol{\epsilon}_t \in \mathbb{R}^r$, $\boldsymbol{\alpha}_t \in \mathbb{R}^p$, $\mathbf{F} \in \mathbb{R}^{p \times p}$ and is invertible, $\mathbf{Q} \in \mathbb{R}^{r \times r}$ is a positive definite matrix, $\boldsymbol{o}_t$ are know offsets, and $\mathbf{R} \in \{0, 1\}^{p \times r}$ with $p \geq r$ contains a subset of the columns of $\mathbf{I}_p$ identity matrix with no duplicate columns in $\mathbf{R}$. The latter implies that $\mathbf{R}^+ = \mathbf{R}^\top$ 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 with r diagonal entries with value 1 and $p - r$ with

value zero. We will let

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

I will use a particle filter and smoother to get smoothed estimates 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 $\mathbf{Q}$ and $\boldsymbol{a}_0$. One choice of smoother is the generalized two-filter smoothing in Fearnhead et al. (2010) and Briers et al. (2010). I have included the $\mathcal{O}(N)$ smoother in Fearnhead et al. (2010) shown in algorithm 1. The rest of vignette is structured as follows: first I cover the particle filter and smoother. Then I cover the EM-algorithm and other miscellaneous topics. I will end with what is implemented at this point.

Considerations

Algorithm 1 shows one of the generalized two-filter smoother from Fearnhead et al. (2010). It requires that we specify the following importance densities and re-sampling weights (optimal values are given as the right hand side):

$$\begin{aligned}
q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) &= \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) \\
\beta_t^{(j)} &\propto \mathrm{P}\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)} \\
\widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \widetilde{\beta}_t^{(k)} &\approx \gamma_t(\boldsymbol{\alpha}_t)\, \mathrm{P}(\boldsymbol{y}_t|\, \boldsymbol{\alpha}_t)\, \mathrm{P}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \boldsymbol{\alpha}_t\right) \frac{\widetilde{w}_{t+1}^{(k)}}{\gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)} \\
\widetilde{q}\left(\widehat{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) &= \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)
\end{aligned} \tag{2}$$

Further, we need to define a backwards filter distribution approximation

$$\widetilde{p}(\boldsymbol{\alpha}_t|\, \boldsymbol{y}_{t:d}) \propto \gamma_t(\boldsymbol{\alpha}_t)\, \mathrm{P}(\boldsymbol{y}_{t:d}|\, \boldsymbol{\alpha}_t) \tag{3}$$

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

Given the models of interest we have that:

- Evaluating $\mathrm{P}(\boldsymbol{y}_t|\, \boldsymbol{\alpha}_t)$ is an expensive operation as $n_{\max} \gg r$ and it has a $\mathcal{O}(n_{\max} r)$ computational cost. Any $\mathcal{O}(n_{\max})$ operation is going to take considerable time.
- Evaluating $f(\boldsymbol{\alpha}_t|\, \boldsymbol{\alpha}_{t-1})$ and $f(\boldsymbol{\alpha}_t|\, \boldsymbol{\alpha}_{t+1})$ is cheap and can be done in closed form and sampling from these distribution can be done in closed form.
- The second example in Fearnhead et al. (2010) is close to the model here though with $n_{\max} = 1$ and $p = 2$.

The following sections will closely follow the example shown in Fearnhead et al. (2010) and the appendix of the paper.

Forward filter (algorithm 2)

This section will cover some options for algorithm 2. Let $\mathcal{N}(\cdot|\cdot,\cdot)$ denote a multivariate normal distribution. We can select the proposal density as

$$q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) = \mathcal{N}\left(\boldsymbol{\xi}_t \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}, \mathbf{Q}\right) \tag{4}$$

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 often called the *bootstrap filter*. Another option is to use normal approximation of $\boldsymbol{y}_t$'s conditional density

$$\begin{aligned} g\left(\boldsymbol{y}_t|\,\boldsymbol{\xi}_t\right) &\simeq \widetilde{g}_t\left(\boldsymbol{y}_t|\,\boldsymbol{\xi}_t\right) \\ &= \mathcal{N}\left(\mathbf{X}_t\boldsymbol{\xi}_t \middle| \mathbf{X}_t\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1} - \mathbf{G}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right)^{-1}\boldsymbol{g}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right), -\mathbf{G}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right)^{-1}\right) \end{aligned} \tag{5}$$

where:

$$\begin{aligned} \mathbf{g}_t\left(\boldsymbol{\xi}\right) &= \left\{ \left.\frac{\partial \log \mathrm{P}(\,y_{it}|\eta_{it})}{\partial \eta_{it}}\right|_{\eta_{it}=\boldsymbol{x}_{it}^\top\boldsymbol{\xi}} \right\}_{i\in R_t} \\ \mathbf{G}_t\left(\boldsymbol{\xi}\right) &= \operatorname{diag}\left(\left\{ \left.\frac{\partial^2 \log \mathrm{P}(\,y_{it}|\eta_{it})}{\partial \eta_{it}^2}\right|_{\eta_{it}=\boldsymbol{x}_{it}^\top\boldsymbol{\xi}} \right\}_{i\in R_t} \right) \end{aligned} \tag{6}$$

to get

$$q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) \propto \mathcal{N}\left(\boldsymbol{\xi}_t \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}, \mathbf{Q}\right) \widetilde{g}\left(\boldsymbol{y}_t \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}\right) \tag{7}$$

Thus, we need to sample from

$$\begin{aligned} q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) &= \mathcal{N}\left(\boldsymbol{\xi}_t|\,\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\right) \\ \boldsymbol{\Sigma}_t^{-1} &= \mathbf{X}_t^\top\left(-\mathbf{G}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right)\right)\mathbf{X}_t + \mathbf{Q}^{-1} \\ \boldsymbol{\mu}_t &= \boldsymbol{\Sigma}_t\left(\mathbf{Q}^{-1}\mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)} + \mathbf{X}_t^\top\left(-\mathbf{G}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right)\right)\left(\mathbf{X}_t\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1} - \mathbf{G}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right)^{-1}\boldsymbol{g}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right)\right)\right) \\ &= \boldsymbol{\Sigma}_t\left(\mathbf{Q}^{-1}\mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)} + \mathbf{X}_t^\top\left(-\mathbf{G}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right)\mathbf{X}_t\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1} + \boldsymbol{g}_t\left(\mathbf{R}^+\mathbf{F}\bar{\boldsymbol{\alpha}}_{t-1}\right)\right)\right) \end{aligned} \tag{8}$$

We can select $\bar{\boldsymbol{\alpha}}_{t-1}$ as the weighted mean given the particle cloud at time $t-1$ (that is, $\{\boldsymbol{\alpha}_{t-1}^{(1)}, \boldsymbol{\alpha}_{t-1}^{(2)}, \ldots \boldsymbol{\alpha}_{t-1}^{(N)}\}$). This is similar to the second order random walk example in Fearnhead et al. (2010). They use the mode which is feasible as the outcome only depends on one element of the state vector. The downside is an $\mathcal{O}\left(n_{\max}p^2 + p^3\right)$ computational cost though independent of the number of particles. The total cost of sampling is $\mathcal{O}\left(n_{\max}p^2 + p^3 + Np^2\right)$. Another option is to set $\bar{\boldsymbol{\alpha}}_{t-1} = \boldsymbol{\alpha}_{t-1}^{(j)}$ for each particle $j = 1, 2, \ldots, N$ in the particle cloud at time $t-1$. This will improve the Taylor expansion but yields an $\mathcal{O}\left(N\left(n_{\max}p^2 + p^3\right)\right)$ computational cost.

Next, we have the re-sampling weights. A simple solution is not to use an auxiliary particle filter as in the examples of Fearnhead et al. (2010) and set:

$$\beta_t^{(j)} \propto w_{t-1}^{(j)} \tag{9}$$

which has an $\mathcal{O}(N)$ cost of sampling. Another options is to set

$$\begin{aligned}
\beta_t^{(j)} &\propto \mathrm{P}\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)} \\
&=\approx w_{t-1}^{(j)} \int_{\mathbb{R}^r} \mathcal{N}\left(\boldsymbol{\xi}_t \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}, \mathbf{Q}\right) g\left(\boldsymbol{y}_t \middle| \boldsymbol{\xi}_t\right) \partial\boldsymbol{\xi}_t \\
&\approx w_{t-1}^{(j)} \int_{\mathbb{R}^r} \mathcal{N}\left(\boldsymbol{\xi}_t \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}, \mathbf{Q}\right) \widetilde{g}_t\left(\boldsymbol{y}_t \middle| \boldsymbol{\xi}_t\right) \partial\boldsymbol{\xi}_t \\
&\approx \frac{w_{t-1}^{(j)} \mathcal{N}\left(\boldsymbol{\mu}_t \middle| \mathbf{R}^+\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}, \mathbf{Q}\right) \widetilde{g}_t\left(\boldsymbol{y}_t \middle| \boldsymbol{\mu}_t\right)}{q\left(\boldsymbol{\mu}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right)}
\end{aligned} \tag{10}$$

where $q\left(\boldsymbol{\mu}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right)$ is meant as the right hand side of the first line in equation (8) computed with $\boldsymbol{\mu}_t$ instead of $\boldsymbol{\xi}_t$. The come at an $\mathcal{O}\left(Np^2\right)$ computational cost assuming that we are using (8) already. Otherwise it has the same computational cost as mentioned when I covered the importance density since we have to make the Taylor expansion approximation.

Backward filter (algorithm 3)

We need to specify the artificial prior $\gamma_t(\boldsymbol{\alpha}_t)$. Briers et al. (2010, page 69 and 70) provides recommendation on the selection. This leads to

$$\begin{aligned}
\gamma_t(\boldsymbol{\alpha}_t) &= \mathcal{N}\left(\boldsymbol{\alpha}_t \middle| \overleftarrow{\boldsymbol{m}}_t, \overleftarrow{\mathbf{P}}_t\right) \\
\overleftarrow{\boldsymbol{m}}_t &= \mathbf{F}^t \boldsymbol{a}_0 \\
\overleftarrow{\mathbf{P}}_t &= \begin{cases} \mathbf{Q}_0 & t = 0 \\ \mathbf{F}\mathbf{P}_{t-1}\mathbf{F}^\top + \mathbf{Q} & t > 0 \end{cases}
\end{aligned} \tag{11}$$

Fearnhead et al. (2010) writes that then we end with

$$\begin{aligned}
\mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t+1}\right) &= \mathcal{N}\left(\boldsymbol{\alpha}_t \middle| \overleftarrow{\boldsymbol{a}}_t, \overleftarrow{\mathbf{S}}_t\right) \\
\overleftarrow{\mathbf{S}}_t &= \overleftarrow{\mathbf{P}}_t \mathbf{F}^\top \overleftarrow{\mathbf{P}}_{t+1}^{-1} \mathbf{Q} \left(\mathbf{F}^\top\right)^{-1} \\
\overleftarrow{\boldsymbol{a}}_t &= \overleftarrow{\mathbf{P}}_t \mathbf{F}^\top \overleftarrow{\mathbf{P}}_{t+1}^{-1} \boldsymbol{\alpha}_{t+1} + \overleftarrow{\mathbf{S}}_t \overleftarrow{\mathbf{P}}_t^{-1} \overleftarrow{\boldsymbol{m}}_t
\end{aligned} \tag{12}$$

which simplifies for the first order random walk to

$$\begin{aligned}
\overleftarrow{\boldsymbol{m}}_t &= \boldsymbol{a}_0 \\
\overleftarrow{\mathbf{P}}_t &= t\mathbf{Q} + \mathbf{Q}_0 \\
\overleftarrow{\mathbf{S}}_t &= (t\mathbf{Q} + \mathbf{Q}_0)\left((t+1)\mathbf{Q} + \mathbf{Q}_0\right)^{-1} \mathbf{Q} \\
\overleftarrow{\boldsymbol{a}}_t &= (t\mathbf{Q} + \mathbf{Q}_0)\left((t+1)\mathbf{Q} + \mathbf{Q}_0\right)^{-1} \boldsymbol{\alpha}_{t+1} + \overleftarrow{\mathbf{S}}_t \overleftarrow{\mathbf{P}}_t^{-1} \overleftarrow{\boldsymbol{m}}_t
\end{aligned} \tag{13}$$

Further, setting $\mathbf{Q}_0 = \mathbf{Q}$ (only in the artificial prior where we may alter γ_0 – see Briers et al. (2010, page 70)) gives us:

$$\begin{aligned} \overleftarrow{\mathbf{S}}_t &= \frac{t+1}{t+2}\mathbf{Q} \\ \overleftarrow{\boldsymbol{a}}_t &= \frac{t+1}{t+2}\boldsymbol{\alpha}_{t+1} + \frac{1}{t+2}\boldsymbol{a}_0 \end{aligned} \tag{14}$$

We get the following backward version of equation (8)

$$\begin{aligned} \widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) &= \mathcal{N}\left(\mathbf{R}^{+}\boldsymbol{\alpha}_t \middle| \overleftarrow{\boldsymbol{\mu}}_t, \overleftarrow{\boldsymbol{\Sigma}}_t\right) \\ \overleftarrow{\boldsymbol{\Sigma}}_t^{-1} &= \mathbf{R}^{+}\overleftarrow{\mathbf{P}}_t^{-1}\mathbf{R}^{+\top} + \mathbf{X}_t^{\top}\left(-\mathbf{G}_t\left(\mathbf{R}^{+}\mathbf{F}^{-1}\bar{\boldsymbol{\alpha}}_{t+1}\right)\right)\mathbf{X}_t + \mathbf{R}^{+}\mathbf{F}^{-1}\mathbf{R}^{+\top}\mathbf{Q}^{-1}\mathbf{R}^{+}\mathbf{F}^{-\top}\mathbf{R}^{+\top} \\ \overleftarrow{\boldsymbol{\mu}}_t &= \overleftarrow{\boldsymbol{\Sigma}}_t\left(\mathbf{R}^{+}\overleftarrow{\mathbf{P}}_t^{-1}\overleftarrow{\boldsymbol{m}}_t + \mathbf{R}^{+}\mathbf{F}^{-1}\mathbf{R}^{+\top}\mathbf{Q}^{-1}\mathbf{R}^{+}\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} + \mathbf{X}_t^{\top}\left(-\mathbf{G}_t\left(\mathbf{R}^{+}\mathbf{F}^{-1}\bar{\boldsymbol{\alpha}}_{t+1}\right)\mathbf{X}_t\mathbf{R}^{+}\mathbf{F}^{-1}\bar{\boldsymbol{\alpha}}_{t+1} + \boldsymbol{g}_t\left(\mathbf{R}^{+}\mathbf{F}^{-1}\bar{\boldsymbol{\alpha}}_{t+1}\right)\right)\right) \end{aligned} \tag{15}$$

which is an approximation of

$$\begin{aligned} \widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) &\propto g\left(\boldsymbol{y}_t \middle| \boldsymbol{\xi}_t\right) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \boldsymbol{\alpha}_t\right) \frac{\gamma_t\left(\boldsymbol{\alpha}_t\right)}{\gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)} \\ &\approx \widetilde{g}\left(\boldsymbol{y}_t \middle| \boldsymbol{\xi}_t\right)\mathcal{N}\left(\widetilde{\gamma}_{t+1}^{(k)} \middle| \mathbf{R}^{+}\mathbf{F}\boldsymbol{\alpha}_t, \mathbf{Q}\right) \frac{\mathcal{N}\left(\boldsymbol{\alpha}_t \middle| \overleftarrow{\boldsymbol{m}}_t, \overleftarrow{\mathbf{P}}_t\right)}{\mathcal{N}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \overleftarrow{\boldsymbol{m}}_{t+1}, \overleftarrow{\mathbf{P}}_{t+1}\right)} \end{aligned} \tag{16}$$

The re-sampling weights can be computed similarly to the forward filter using the backward transition density in equation (12) in the numerator. We can also use a bootstrap like filter with the importance density by sampling from (12).

Combining / smoothing (algorithm 1)

We need to specify the importance density $\widetilde{q}\left(\cdot \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)$ (see Fearnhead et al. (2010, page 453)). Again, we can use a idea similar to those in the appendix of Fearnhead et al. (2010) and choose

$$\begin{aligned} \widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) &= \mathcal{N}\left(\boldsymbol{\xi}_t \middle| \overrightarrow{\boldsymbol{\mu}}_t, \overrightarrow{\boldsymbol{\Sigma}}_t\right) \\ \overrightarrow{\boldsymbol{\Sigma}}_t^{-1} &= \mathbf{Q}^{-1} + \mathbf{X}_t^{\top}\left(-\mathbf{G}_t\left(\bar{\boldsymbol{\xi}}_t\right)\right)\mathbf{X}_t + \mathbf{R}^{+}\mathbf{F}^{-1}\mathbf{R}^{+\top}\mathbf{Q}^{-1}\mathbf{R}^{+}\mathbf{F}^{-\top}\mathbf{R}^{+\top} \\ \overrightarrow{\boldsymbol{\mu}}_t &= \overrightarrow{\boldsymbol{\Sigma}}_t\left(\mathbf{Q}^{-1}\mathbf{R}^{+}\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)} + \mathbf{R}^{+}\mathbf{F}^{-1}\mathbf{R}^{+\top}\mathbf{Q}^{-1}\mathbf{R}^{+}\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} + \mathbf{X}_t^{\top}\left(-\mathbf{G}_t\left(\bar{\boldsymbol{\xi}}_t\right)\mathbf{X}_t\bar{\boldsymbol{\xi}}_t + \boldsymbol{g}_t\left(\bar{\boldsymbol{\xi}}_t\right)\right)\right) \end{aligned} \tag{17}$$

where $\bar{\boldsymbol{\xi}}_t$ can be a combined mean given the cloud means at time $t-1$ and $t+1$ or a mean for each of the two drawn particles in the (j_i, k_i) pairs. This is to approximate

$$\widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \propto \widetilde{g}\left(\boldsymbol{y}_t \middle| \boldsymbol{\xi}_t\right) f\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) \frac{f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \boldsymbol{\alpha}_t\right)}{\gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)} \tag{18}$$

We can also use a bootstrap like filter by sampling from

$$\begin{aligned}
\widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) &= \mathcal{N}\left(\boldsymbol{\xi}_t \middle| \widetilde{\boldsymbol{m}}, \widetilde{\mathbf{S}}\right) \\
\widetilde{\mathbf{S}} &= \left(\mathbf{Q}^{-1} + \mathbf{R}^{+}\mathbf{F}^{-1}\mathbf{R}^{+\top}\mathbf{Q}^{-1}\mathbf{R}^{+}\mathbf{F}^{-\top}\mathbf{R}^{+\top}\right)^{-1} \\
\widetilde{\boldsymbol{m}} &= \widetilde{\mathbf{S}}\left(\mathbf{Q}^{-1}\mathbf{R}^{+}\mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)} + \mathbf{R}^{+}\mathbf{F}^{-1}\mathbf{R}^{+\top}\mathbf{Q}^{-1}\mathbf{R}^{+}\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)
\end{aligned} \tag{19}$$

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

Input:

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

Importance density which optimally is (see Fearnhead et al. (2010, page 453)):

$$\widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) = \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \tag{20}$$

Let $\boldsymbol{\alpha}_t^{(i)}$ denote particle i at time t, $w_t^{(i)}$ denote the weight of the particle and $\beta_t^{(i)}$ denote the re-sampling weight.

procedure FILTER FORWARD

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

procedure FILTER BACKWARDS

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

procedure SMOOTH (COMBINE)

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

Re-sample:

$i = 1, 2, \ldots, N_s$ pairs of (j_i, k_i) 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 importance density $\widetilde{q}\left(\cdot \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right)$.

Re-weight:

Assign each particle weights:

$$\widehat{w}_t^{(i)} \propto \frac{\mathrm{P}\left(\widehat{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}\right) \mathrm{P}\left(\boldsymbol{y}_t \middle| \widehat{\boldsymbol{\alpha}}_t^{(i)}\right) \mathrm{P}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)} \middle| \widehat{\boldsymbol{\alpha}}_t^{(i)}\right) w_{t-1}^{(j_i)} \widetilde{w}_{t+1}^{(k_i)}}{\widetilde{q}\left(\widehat{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right) \beta_t^{(j_i)} \widetilde{\beta}_t^{(k_i)} \gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right)} \tag{21}$$

Algorithm 2 Forward filter due to Pitt and Shephard (1999). You can compare with Doucet and Johansen (2009, page 20 and 25). The version and notation below is from Fearnhead et al. (2010, page 449).

Input:

Importance density and specification of weights are optimally given by:

$$\begin{gathered} q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) = \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) \\ \beta_t^{(j)} \propto \mathrm{P}\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) w_{t-1}^{(j)} \end{gathered} \tag{22}$$

1: Sample $\boldsymbol{\alpha}_0^{(1)}, \ldots, \boldsymbol{\alpha}_0^{(N_f)}$ particles from $\mathcal{N}\left(\cdot \middle| \boldsymbol{a}_0, \mathbf{Q}_0\right)$ and set the weights $w_0^{(1)}, \ldots, w_0^{(N_f)}$ to $1/N_f$.
2: **for** $t = 1, \ldots, d$ **do**
3: **procedure** RE-SAMPLE
4: Compute re-sampling weights $\beta_t^{(j)}$ and re-sample according to $\beta_t^{(j)}$ to get indices $j_1, \ldots j_N$.
5: **procedure** PROPAGATE
6: Sample new particles $\boldsymbol{\alpha}_t^{(i)}$ using importance density $q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t\right)$.
7: **procedure** RE-WEIGHT
8: Re-weight particles using:

$$w_t^{(i)} \propto \frac{\mathrm{P}\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_t^{(i)}\right) \mathrm{P}\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}\right) w_{t-1}^{(j_i)}}{q\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t\right) \beta_t^{(j_i)}} \tag{23}$$

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

Input:

A backwards filter distribution approximation:

$$\widetilde{p}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{t:d}\right) \propto \gamma_t\left(\boldsymbol{\alpha}_t\right) \mathrm{P}\left(\boldsymbol{y}_{t:d} \middle| \boldsymbol{\alpha}_t\right) \tag{24}$$

with an artificial prior distribution $\gamma_t\left(\boldsymbol{\alpha}_t\right)$. This is introduced as $\mathrm{P}\left(\boldsymbol{y}_{t:d} \middle| \boldsymbol{\alpha}_t\right)$ is not a density function in $\boldsymbol{\alpha}_t$.

Importance density and specification of weights (Fearnhead et al. (2010, page 451 – look in the example in the appendix)):

$$\widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \widetilde{\beta}_t^{(k)} \approx \gamma_t\left(\boldsymbol{\alpha}_t\right) \mathrm{P}\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_t\right) \mathrm{P}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \boldsymbol{\alpha}_t\right) \frac{\widetilde{w}_{t+1}^{(k)}}{\gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)} \tag{25}$$

where we want (see Briers et al. (2010, page 74)):

$$\begin{aligned} \widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) &\propto \mathrm{P}\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_t\right) \mathrm{P}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \boldsymbol{\alpha}_t\right) \frac{\gamma_t(\boldsymbol{\alpha}_t)}{\gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)} \\ \widetilde{\beta}_t^{(k)} &\propto \widetilde{p}\left(\boldsymbol{y}_t \middle| \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \widetilde{w}_{t+1}^{(k)} \end{aligned} \tag{26}$$

1: Sample $\widetilde{\boldsymbol{\alpha}}_{d+1}^{(1)}, \ldots, \widetilde{\boldsymbol{\alpha}}_{d+1}^{(N_f)}$ particles from $\gamma_{d+1}(\cdot)$ and set the weights $\widetilde{w}_{d+1}^{(1)}, \ldots, w_{d+1}^{(N_f)}$ to $1/N_f$.
2: **for** $t = d, \ldots, 1$ **do**
3: **procedure** RE-SAMPLE
4: Compute re-sampling weights $\widetilde{\beta}_t^{(k)}$ and re-sample according to $\widetilde{\beta}_t^{(k)}$ to get indices $k_1, \ldots k_N$.
5: **procedure** PROPAGATE
6: Sample new particles $\widetilde{\boldsymbol{\alpha}}_t^{(i)}$ using importance density $\widetilde{q}\left(\boldsymbol{\alpha}_t \middle| \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}, \boldsymbol{y}_t\right)$.
7: **procedure** RE-WEIGHT
8: Re-weight particles using (see Briers et al. (2010, page 72) and add the ratio of prior probability and re-sampling weight):

$$\widetilde{w}_t^{(i)} \propto \frac{\mathrm{P}\left(\boldsymbol{y}_t \middle| \widetilde{\boldsymbol{\alpha}}_t^{(i)}\right) \mathrm{P}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)} \middle| \widetilde{\boldsymbol{\alpha}}_t^{(i)}\right) \widetilde{w}_{t+1}^{(k_i)} \gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)}{q\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}, \boldsymbol{y}_t\right) \beta_t^{(k_i)} \gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right)} \tag{27}$$

2 Log likelihood evaluation

We can evaluate the log likelihood for a particular value of $\boldsymbol{\theta} = \{\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{a}_0, \mathbf{F}\}$ as described in Doucet and Johansen (2009, page 5) and Malik and Pitt (2011, page 193) using the forward particle filter shown in algorithm 2.

3 Parameter estimation

This section shows an example of parameter estimation in a random walk. The formulas for parameter estimation with a given order random with the EM-algorithm (Dempster et al., 1977) are particularly simple. We need to estimate $\mathbf{Q}$ and $\boldsymbol{a}_0$ elements of $\boldsymbol{\theta} = \{\mathbf{Q}, \mathbf{Q}_0, \boldsymbol{a}_0, \mathbf{F}\}$. We do this by running algorithm 1 for the current $\boldsymbol{\theta}$. Then we compute the so-called *smoothed additive functional* or summary statistics

$$
\begin{aligned}
\boldsymbol{t}_t^{(\boldsymbol{\theta})} &= \int_{\boldsymbol{\alpha}_t} \boldsymbol{\alpha}_t \mathrm{P}_{\boldsymbol{\theta}} \left(\boldsymbol{\alpha}_t \middle| \, \boldsymbol{y}_{1:d} \right) \partial \boldsymbol{\alpha}_t \\
&\approx \sum_{i=1}^{N_s} \widehat{\boldsymbol{\alpha}}_t^{(i)} \widehat{w}_t^{(i)} \approx \sum_{i=1}^{N_s} \boldsymbol{\alpha}_t^{(j_i)} \widehat{w}_{t+1}^{(i)} \approx \sum_{i=1}^{N_s} \widetilde{\boldsymbol{\alpha}}_t^{(k_i)} \widehat{w}_{t-1}^{(i)} \\
\boldsymbol{T}_t^{(\boldsymbol{\theta})} &= \int_{\mathbb{R}^{2p}} \left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1} \right) \left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1} \right)^\top \mathrm{P}_{\boldsymbol{\theta}} \left(\boldsymbol{\alpha}_{(t-1):t} \middle| \, \boldsymbol{y}_{1:d} \right) \partial \boldsymbol{\alpha}_{(t-1):t} \\
&\approx \sum_{i=1}^{N_s} \left(\widehat{\boldsymbol{\alpha}}_t^{(i)} - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j_i)} \right) \left(\widehat{\boldsymbol{\alpha}}_t^{(i)} - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j_i)} \right)^\top \widehat{w}_t^{(i)} \\
&\approx \sum_{i=1}^{N_s} \left(\widetilde{\boldsymbol{\alpha}}_t^{(k_i)} - \mathbf{F}\widehat{\boldsymbol{\alpha}}_{t-1}^{(i)} \right) \left(\widetilde{\boldsymbol{\alpha}}_t^{(k_i)} - \mathbf{F}\widehat{\boldsymbol{\alpha}}_{t-1}^{(i)} \right)^\top \widehat{w}_{t-1}^{(i)}
\end{aligned}
\tag{28}
$$

where $\boldsymbol{\alpha}_{s:t} = \{\boldsymbol{\alpha}_s, \boldsymbol{\alpha}_{s+1}, \dots \boldsymbol{\alpha}_t\}$ and the subscript in P denotes that it is the probability given the parameter $\boldsymbol{\theta}$. I use that the normalized weights $\widehat{w}_t^{(i)}$ and the pairs $\left\{ \boldsymbol{\alpha}_{t-1}^{(j_i)}, \widehat{\boldsymbol{\alpha}}_t^{(i)}, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)} \right\}$ form a discrete approximation of $\mathrm{P}_{\boldsymbol{\theta}} \left(\boldsymbol{\alpha}_{(t-1):(t+1)} \middle| \, \boldsymbol{y}_{1:d} \right)$. That is,

$$
\begin{aligned}
&\mathrm{P}_{\boldsymbol{\theta}} \left(\boldsymbol{\alpha}_{(t-1):t+1} \middle| \, \boldsymbol{y}_{1:d} \right) \\
&\qquad \approx \sum_{i=1}^{N_s} \widehat{w}_t^{(i)} \delta \left(\boldsymbol{\alpha}_{t-1} - \boldsymbol{\alpha}_{t-1}^{(j_i)} \right) \delta \left(\boldsymbol{\alpha}_t - \widehat{\boldsymbol{\alpha}}_t^{(i)} \right) \delta \left(\boldsymbol{\alpha}_{t+1} - \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)} \right)
\end{aligned}
\tag{29}
$$

where $\delta(\cdot)$ is the Dirac delta function. The update of $\boldsymbol{a}_0$ and $\mathbf{Q}$ given the summary statistics is

$$
\boldsymbol{a}_0 = \boldsymbol{t}_0^{(\boldsymbol{\theta})} \qquad \mathbf{Q} = \frac{1}{d} \sum_{t=1}^{d} \mathbf{R}^{+} \boldsymbol{T}_t^{(\boldsymbol{\theta})} \mathbf{R}^{+\top} \tag{30}
$$

We then repeat with the new $\boldsymbol{a}_0$ and $\mathbf{Q}$ for a given number of iterations or till a convergence criteria is satisfied. See Kantas et al. (2015), Del Moral et al.

(2010) and Schön et al. (2011) for further details on parameter estimation with particle filters.

We can do parts of the probability estimates exact at time 1 by using

$$\begin{aligned}
&\mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_0 \middle| \boldsymbol{y}_{1:d}\right) \\
&\quad = \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_0 \middle| \boldsymbol{\alpha}_1, \boldsymbol{y}_{1:d}\right) \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1 \middle| \boldsymbol{y}_{1:d}\right) \\
&\quad = \frac{\mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1, \boldsymbol{y}_{1:d} \middle| \boldsymbol{\alpha}_0\right) \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_0\right)}{\mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1, \boldsymbol{y}_{1:d}\right)} \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1 \middle| \boldsymbol{y}_{1:d}\right) \\
&\quad = \frac{\mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{1:d} \middle| \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_0\right) f_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1 \middle| \boldsymbol{\alpha}_0\right) \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_0\right)}{\mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1\right) \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{1:d} \middle| \boldsymbol{\alpha}_1\right)} \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1 \middle| \boldsymbol{y}_{1:d}\right) \\
&\quad = \frac{f_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1 \middle| \boldsymbol{\alpha}_0\right) \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_0\right)}{\mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1\right)} \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1 \middle| \boldsymbol{y}_{1:d}\right) \\
&\quad = \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_0 \middle| \boldsymbol{\alpha}_1\right) \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1 \middle| \boldsymbol{y}_{1:d}\right) \\
&\quad = \mathcal{N}\left(\boldsymbol{\alpha}_0 \middle| \boldsymbol{m}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1\right), \mathbf{S}_{\boldsymbol{\theta}}\right) \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}_1 \middle| \boldsymbol{y}_{1:d}\right) \\
&\quad \approx \sum_{i=1}^{N_s} \mathcal{N}\left(\boldsymbol{\alpha}_0 \middle| \boldsymbol{m}_{\boldsymbol{\theta}}\left(\widehat{\boldsymbol{\alpha}}_1^{(i)}\right), \mathbf{S}_{\boldsymbol{\theta}}\right) \widehat{w}_t^{(i)} \delta\left(\boldsymbol{\alpha}_1 - \widehat{\boldsymbol{\alpha}}_1^{(i)}\right)
\end{aligned} \tag{31}$$

where we use that $\mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{1:d} \middle| \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_0\right) = \mathrm{P}_{\boldsymbol{\theta}}\left(\boldsymbol{y}_{1:d} \middle| \boldsymbol{\alpha}_1\right)$ and let

$$\begin{aligned}
\mathbf{S}_{\boldsymbol{\theta}}^{-1} &= \mathbf{Q}_0^{-1} + \mathbf{F}^{-1}\mathbf{R}\mathbf{Q}^{-1}\mathbf{R}^\top\mathbf{F}^{-\top} \\
\boldsymbol{m}_{\boldsymbol{\theta}}\left(\boldsymbol{\alpha}\right) &= \mathbf{S}_{\boldsymbol{\theta}}\left(\mathbf{Q}_0^{-1}\boldsymbol{a}_0 + \mathbf{F}^{-1}\mathbf{R}\mathbf{Q}^{-1}\mathbf{R}^{+}\boldsymbol{a}\right)
\end{aligned} \tag{32}$$

Thus, we end with

$$\begin{aligned}
\boldsymbol{T}_1^{(\boldsymbol{\theta})} \approx \sum_{i=1}^{N_s} \widehat{w}_1^{(i)} \Bigg(& \widehat{\boldsymbol{\alpha}}_1^{(i)} \left(\widehat{\boldsymbol{\alpha}}_1^{(i)}\right)^\top \\
& - \mathbf{F}\boldsymbol{m}_{\boldsymbol{\theta}}\left(\widehat{\boldsymbol{\alpha}}_1^{(i)}\right)\left(\widehat{\boldsymbol{\alpha}}_1^{(i)}\right)^\top - \widehat{\boldsymbol{\alpha}}_1^{(i)}\boldsymbol{m}_{\boldsymbol{\theta}}\left(\widehat{\boldsymbol{\alpha}}_1^{(i)}\right)^\top\mathbf{F}^\top \\
& + \mathbf{F}\boldsymbol{m}_{\boldsymbol{\theta}}\left(\widehat{\boldsymbol{\alpha}}_1^{(i)}\right)\boldsymbol{m}_{\boldsymbol{\theta}}\left(\widehat{\boldsymbol{\alpha}}_1^{(i)}\right)^\top\mathbf{F}^\top \Bigg) + \mathbf{S}_{\boldsymbol{\theta}}
\end{aligned} \tag{33}$$

4 Other 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 the article. For instance, only the Taylor expansion approximation around a single point to approximate g would be feasible. The non-auxiliary version in Briers et al. (2010) is more feasible as it only requires evaluation of f in the smoothing part of two-filter smoother (see equation (46) in the 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. (2010) and Del Moral et al. (2010) they are still worthy candidates as the computational cost is independent of the number of observations, n. Further, the computational cost can be reduced to $\mathcal{O}\left(N\log(N)\right)$ with the approximations 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 evaluation. I am not sure how well these method scale with higher state dimension, p.

Kantas et al. (2015) show empirically that it may be worth just using a forward filter. However, the example is with a univariate outcome ($n = 1$ – not to be confused with the number of periods d). The cost here of the forward filter is at least $\mathcal{O}\left(dNn_{\max}p\right)$. Every new particle yields an $\mathcal{O}\left(dn_{\max}p\right)$ cost which is expensive due to the large number of outcomes, n. Thus, the considerations are different and a $\mathcal{O}\left(dNn_{\max}p + N^2\right)$ method will not make a big difference unless N is large. Another alternative is to add noise to $\boldsymbol{\theta}_t$ and use the methods in Andrieu and Doucet (2002).

5 Briers et al. (2010)

The $\mathcal{O}\left(N^2\right)$ smother from Briers et al. (2010) 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 (37) comes from the two-filter formula:

$$\begin{aligned}
\mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{1:d}\right) &= \frac{\mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{1:t-1}\right)\mathrm{P}\left(\boldsymbol{y}_{t:d} \middle| \boldsymbol{\alpha}_t\right)}{\mathrm{P}\left(\boldsymbol{y}_{t:d} \middle| \boldsymbol{y}_{1:t-1}\right)} \\
&\propto \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{1:t-1}\right)\mathrm{P}\left(\boldsymbol{y}_{t:d} \middle| \boldsymbol{\alpha}_t\right) \\
&= \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{1:t-1}\right)\frac{\mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{t:d}\right)\mathrm{P}\left(\boldsymbol{y}_{t:d}\right)}{\mathrm{P}\left(\boldsymbol{\alpha}_t\right)} \\
&\propto \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{1:t-1}\right)\frac{\mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{t:d}\right)}{\mathrm{P}\left(\boldsymbol{\alpha}_t\right)} \\
&= \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{t:d}\right)\frac{\left[\int_{\boldsymbol{\alpha}_{t-1}} \mathrm{P}\left(\boldsymbol{\alpha}_{t-1} \middle| \boldsymbol{y}_{1:t-1}\right) f\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}\right) \partial\boldsymbol{\alpha}_{t-1}\right]}{\mathrm{P}\left(\boldsymbol{\alpha}_t\right)} \\
&\approx \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta\left(\boldsymbol{\alpha}_t - \widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)\frac{\left[\sum_{j=1}^{N} w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]}{\gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)}
\end{aligned} \tag{34}$$

Similar arguments leads to:

$$
\begin{aligned}
&\mathrm{P}\left(\boldsymbol{\alpha}_t, \boldsymbol{\alpha}_{t-1} \middle| \boldsymbol{y}_{1:d}\right) \\
&\quad \propto \mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{t:d}\right) \frac{\mathrm{P}\left(\boldsymbol{\alpha}_{t-1} \middle| \boldsymbol{y}_{1:t-1}\right) f\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}\right)}{\mathrm{P}\left(\boldsymbol{\alpha}_t\right)} \\
&\quad \approx \sum_{i=1}^{N} \sum_{j=1}^{N} \widetilde{w}_t^{(i)} \delta\left(\boldsymbol{\alpha}_t - \widetilde{\boldsymbol{\alpha}}_t^{(i)}\right) \frac{\left[\sum_{j=1}^{N} w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]}{\gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)} \frac{w_{t-1}^{(j)} \delta\left(\boldsymbol{\alpha}_{t-1} - \boldsymbol{\alpha}_{t-1}^{(j)}\right) f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)}{\left[\sum_{j=1}^{N} w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]} \\
&\quad = \sum_{i=1}^{N} \sum_{j=1}^{N} \widehat{w}_t^{(i,j)} \delta\left(\boldsymbol{\alpha}_t - \widetilde{\boldsymbol{\alpha}}_t^{(i)}\right) \delta\left(\boldsymbol{\alpha}_{t-1} - \boldsymbol{\alpha}_{t-1}^{(j)}\right)
\end{aligned} \tag{35}
$$

where

$$
\widehat{w}_t^{(i,j)} = \widehat{w}_t^{(i)} \frac{w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)}{\left[\sum_{j=1}^{N} w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]} \tag{36}
$$

The above is what we need for the EM-algorithm.

Algorithm 4 $\mathcal{O}\left(N^2\right)$ generalized two filter smoother using the method in Briers et al. (2010).

Input:

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

1: **procedure** FILTER FORWARD

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

3: **procedure** FILTER BACKWARDS

4: 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 $\mathrm{P}\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{y}_{t:d}\right)$ for $t = d+1, d, d-1, \ldots, 1$. See algorithm 3.

5: **procedure** SMOOTH (COMBINE)

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

7: Assign each backward filter particle a smoothing weight given by:

$$
\widehat{w}_t^{(i)} \propto \widetilde{w}_t^{(i)} \frac{\left[\sum_{j=1}^{N} w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]}{\gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)} \tag{37}
$$

---

6 Implementation

The `PF_EM` method in the `dynamichazard` package contains an implementation of the above described method. 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 particle in the smoothing step, $N_s > N$, as pointed out in the discussion in Fearnhead et al. (2010, page 460 and 461). Further, selecting $N_f > N$ may be preferable to ensure coverage of the state space at time 0 and $d + 1$.

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

- `"bootstrap_filter"` for a bootstrap filter.
- `"PF_normal_approx_w_cloud_mean"` and `"AUX_normal_approx_w_cloud_mean"` for the Taylor expansion normal approximation of the conditional density of $\boldsymbol{y}_t$ made around the weighted mean of the previous cloud. 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 expansion normal approximation of the conditional density of $\boldsymbol{y}_t$ made around the parent (or/and child) particle. 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 Resampling* (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.

6.1 Linear maps

The methods describe above involves many linear maps. These are implemented with `C++` abstract class with specialized `map` member functions for particular problem to decrease the computation. An alternative would have been to use a sparse matrix implementation. A.s. an example we have a mapping matrix $\mathbf{A}$ which `C++` abstract member on the main data object used in the package called `xyz`. E.g., this could $\mathbf{F}$ with name `state_trans`. Then the following operations are implemented

- `map()`: Returns $\mathbf{A}$.
- `map(const arma::vec &x, bool tranpose)`: Returns $\mathbf{A}^\top \boldsymbol{x}$ if `tranpose == true` and $\mathbf{A}\boldsymbol{x}$ otherwise.
- `map(const arma::mat &X, side s, bool tranpose)`: Let $\mathbf{B} = \mathbf{A}^\top$ if `tranpose == true` and otherwise $\mathbf{B} = \mathbf{A}$. Then the result is $\mathbf{B}\mathbf{X}\mathbf{B}^\top$ if `s == both`, $\mathbf{B}\mathbf{X}$ if `s == right`, and $\mathbf{X}\mathbf{B}^\top$ if `s == left`.

These are implemented for

C++ member name	Matrix $\mathbf{A}$
`err_state`	$\mathbf{R}$
`err_state_inv`	$\mathbf{R}^{+} = \mathbf{R}^{\top}$
`state_trans`	$\mathbf{F}$
`state_trans_err`	$\mathbf{R}^{+}\mathbf{F} = \mathbf{R}^{\top}\mathbf{F}$
`state_trans_inv`	$\mathbf{F}^{-1}$
`state_trans_inv_err`	$\mathbf{R}^{+}\mathbf{F}^{-1}\mathbf{R}^{+\top} = \mathbf{R}^{\top}\mathbf{F}^{-1}\mathbf{R}$

Further, we will need function to compute terms $\mathbf{R}^{+}\overleftarrow{\mathbf{P}}_{t}^{-1}\overleftarrow{\boldsymbol{m}}_{t}$ and $\mathbf{R}^{+}\mathbf{P}_{t}^{-1}\mathbf{R}$ for equation (15). This is done with the virtual method `uncond_mean` and `uncond_covar`. These will computed once using equation (12).

The motivation for making functions for these is e.g., in the dense matrix case we reduce the computational cost of $\mathbf{R}^{+}\mathbf{F}\mathbf{R}^{+\top}\mathbf{Z}$ to $\mathcal{O}\left(r^3\right)$ from $\mathcal{O}\left(p^3\right)$. Further, since the methods are virtual, then we can specialize for specific types of model. E.g., the first order random walk has

C++ member name	Matrix $\mathbf{A}$
`err_state`	$\mathbf{R} = \mathbf{I}_r$
`err_state_inv`	$\mathbf{R}^{+} = \mathbf{I}_r$
`state_trans`	$\mathbf{F} = \mathbf{I}_r$
`state_trans_err`	$\mathbf{R}^{+}\mathbf{F} = \mathbf{I}_r$
`state_trans_inv`	$\mathbf{F}^{-1} = \mathbf{I}_r$
`state_trans_inv_err`	$\mathbf{R}^{+}\mathbf{F}^{-1}\mathbf{R}^{+\top} = \mathbf{I}_r$

where $\mathbf{I}_r$ is an r dimensional identity matrix. Thus, there is a $\mathcal{O}\left(1\right)$ cost of all the above operations.

References

Christophe Andrieu and Arnaud Doucet. Particle filtering for partially observed gaussian state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):827–836, 2002.

Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1): 61–89, 2010.

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.

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. 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.

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

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

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