Particle filters in the dynamichazard package

Benjamin Christoffersen

December 25, 2018

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 model is

$$\begin{aligned} y_{it} &\sim p\left(y_{it}|\eta_{it}\right) \\ \boldsymbol{\eta}_t &= \mathbf{X}_t\mathbf{R}^+\boldsymbol{\alpha}_t + \boldsymbol{o}_t + \mathbf{Z}_t\boldsymbol{\omega} \\ \boldsymbol{\alpha}_t &= \mathbf{F}\boldsymbol{\alpha}_{t-1} + \mathbf{R}\boldsymbol{\epsilon}_t \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\right)' \\ &\qquad\qquad \boldsymbol{\alpha}_0 \sim \mathcal{N}\left(\boldsymbol{\mu}_0, \mathbf{Q}_0\right) \end{aligned} \qquad \begin{aligned} &i = 1, \dots, n_t \\ &t = 1, \dots, d \end{aligned} \tag{1}$$

where I denote the conditional densities as $g_t\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right) = g\left(\boldsymbol{y}_t|\mathbf{X}_t\mathbf{R}^+\boldsymbol{\alpha}_t + \boldsymbol{o}_t + \mathbf{Z}_t\boldsymbol{\omega}\right)$ and $f\left(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}\right)$. We are in a survival analysis setting where the simplest model has an indicator of an event of individual i in time t such that $y_{it} \in \{0, 1\}$, η_{it} is the linear predictor, and we use the logistic function as the link function between η_{it} and the probability of an event. For each $t = 1, \dots, d$, we a have risk set given by $R_t \subseteq \{1, 2, \dots, n\}$. Further, we let $n_t = |R_t|$ 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 fixed effects and $\boldsymbol{\omega}$ are the corresponding coefficients.

Superscript + denotes the Moore-Penrose inverse, the i'th row of $\mathbf{X}_t$ is $\boldsymbol{x}_{it}$, $\boldsymbol{x}_{it}, \boldsymbol{\epsilon}_t \in \mathbb{R}^r$, $\boldsymbol{\mu}_0, \boldsymbol{\alpha}_t \in \mathbb{R}^p$, $\mathbf{F} \in \mathbb{R}^{p\times p}$, $\mathbf{Q} \in \mathbb{R}^{r\times r}$ is a positive definite matrix, $\mathbf{Q}_0 \in \mathbb{R}^{p\times p}$ is a positive definite matrix, $\boldsymbol{o}_t$ are known 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 order the entires of $\mathbf{R}$ such that the first r columns are the first r columns of the identity matrix. I.e.,

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

The data sets we are working with have $n_{\max} \gg p \geq r$ (e.g. $n_{\max} = 100000$ 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 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 $\mathbf{Q}$, $\boldsymbol{\omega}$, and $\boldsymbol{\mu}_0$. One choice of smoother is the generalized two-filter smoother in Fearnhead et al. (2010) and Briers et al. (2009). The rest of vignette is structured as follows: first I cover the particle filter and smoother in Fearnhead et al. (2010). Then I cover the EM-algorithm, smoother in Briers et al. (2009), and other miscellaneous topics. I will end with some points about the implementation.

1.1 Proposal distributions and re-sampling weights

Algorithm 1 shows one of the generalized two-filter smoother from Fearnhead et al. (2010) in the first order state space model. This is the situation where $r = p$ 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 the model where

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

for some known positive definite matrix $\mathbf{H}_t$. This is not implemented but deriving optimal re-sampling weights and proposal distributions is possible in this case. Thus, this will turn out to be useful to motivate the approximations we use later. The state space model is

$$\begin{aligned}
\boldsymbol{y}_t &\sim \mathcal{N}\left(\boldsymbol{\eta}_t, \mathbf{H}_t\right) \\
\boldsymbol{\eta}_t &= \mathbf{X}_t\boldsymbol{\alpha}_t + \boldsymbol{o}_t + \mathbf{Z}_t\boldsymbol{\omega} && && i = 1, \dots, n_t \\
\boldsymbol{\alpha}_t &= \mathbf{F}\boldsymbol{\alpha}_{t-1} + \mathbf{R}\boldsymbol{\epsilon}_t && \boldsymbol{\epsilon}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\right)' && t = 1, \dots, d \\
& && \boldsymbol{\alpha}_0 \sim \mathcal{N}\left(\boldsymbol{\mu}_0, \mathbf{Q}_0\right)
\end{aligned}$$

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

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

where $\phi\left(\cdot | \boldsymbol{m}, \mathbf{M}\right)$ is the density function of a multivariate normal distribution with mean $\boldsymbol{m}$ and covariance matrix $\mathbf{M}$. 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

$$\begin{aligned}
\log p\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t\right) &= \log p\left(\boldsymbol{\alpha}_t, \boldsymbol{y}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) + \dots \\
&= \log g_t\left(\boldsymbol{y}_t | \boldsymbol{\alpha}_t\right) + \log f\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right) + \dots \\
&= -\frac{1}{2}\left(\boldsymbol{y}_t - \mathbf{X}_t\boldsymbol{\alpha}_t - \boldsymbol{h}_t\right)^\top \mathbf{H}_t^{-1}\left(\boldsymbol{y}_t - \mathbf{X}_t\boldsymbol{\alpha}_t - \boldsymbol{h}_t\right) \\
&\quad - \frac{1}{2}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}\right)^\top \mathbf{Q}^{-1}\left(\boldsymbol{\alpha}_t - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j)}\right) + \dots \\
&= -\frac{1}{2}\boldsymbol{\alpha}_t^\top \boldsymbol{\Sigma}_t^{-1}\boldsymbol{\alpha}_t + \boldsymbol{\alpha}_t^\top \boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}\left(\boldsymbol{\alpha}_{t-1}^{(j)}\right) + \dots \\
\boldsymbol{\Sigma}_t &= \left(\mathbf{Q}^{-1} + \mathbf{X}_t^\top \mathbf{H}_t^{-1}\mathbf{X}_t\right)^{-1} \qquad (3) \\
\boldsymbol{\mu}(\boldsymbol{x}) &= \boldsymbol{\Sigma}_t\left(\mathbf{Q}^{-1}\mathbf{F}\boldsymbol{x} + \mathbf{X}_t^\top \mathbf{H}_t^{-1}(\boldsymbol{y}_t - \boldsymbol{h}_t)\right) \qquad (4)
\end{aligned}$$

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

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

We can also let proposal density be

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

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 often called the *bootstrap filter*. This is 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 $\gamma_t\left(\boldsymbol{\alpha}_t\right)$. Briers et al. (2009, page 69 and 70) provides recommendation on the selection. One suggestion is the artificial density function

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

We use this artificial prior to define a distribution with the following conditional density functions

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

where we have left out some of the normalization constants. Sampling from this artificial distribution turns out to be useful. To derive the re-sampling weight, we first find an expression for the density $\widetilde{p}\left(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t+1}\right)$. We can observe that

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

so

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

.

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

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

The first equality can be shown by

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

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

$$\begin{aligned}\widetilde{\beta}_t^{(k)} &\propto \widetilde{p}\left(\boldsymbol{y}_t\middle|\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)\widetilde{w}_{t+1}^{(k)} \\ &\propto \int g_t\left(\boldsymbol{y}_i|\boldsymbol{\alpha}_t\right)\widetilde{p}\left(\boldsymbol{\alpha}_t\middle|\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)d\boldsymbol{\alpha}_t\widetilde{w}_{t+1}^{(k)} \\ &= \phi\left(\boldsymbol{y}_t\middle|\mathbf{X}_t\overleftarrow{\boldsymbol{a}}_t(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}) + \boldsymbol{h}_t, \mathbf{X}_t\overleftarrow{\mathbf{S}}_t\mathbf{X}_t^\top + \mathbf{H}_t\right)\widetilde{w}_{t+1}^{(k)}\end{aligned} \tag{9}$$

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

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

Thus, we set

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

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

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

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

$$\begin{aligned}&\log q\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)},\boldsymbol{y}_t,\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \\ &\quad= \log g\left(\boldsymbol{y}_t|\boldsymbol{\alpha}_t\right)+\log f\left(\boldsymbol{\alpha}_t\middle|\boldsymbol{\alpha}_{t-1}^{(j)}\right)+\log f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\middle|\boldsymbol{\alpha}_t\right)+\dots \\ &\quad= -\frac{1}{2}\boldsymbol{\alpha}_t^\top\overleftrightarrow{\boldsymbol{\Sigma}}_t^{-1}\boldsymbol{\alpha}_t+\boldsymbol{\alpha}_t^\top\overleftrightarrow{\boldsymbol{\Sigma}}_t^{-1}\overleftrightarrow{\boldsymbol{\mu}}(\boldsymbol{\alpha}_{t-1}^{(j)},\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)})+\dots \\ \overleftrightarrow{\boldsymbol{\Sigma}}_t &= \left(\mathbf{Q}^{-1}+\mathbf{F}^\top\mathbf{Q}^{-1}\mathbf{F}+\mathbf{X}_t^\top\mathbf{H}^{-1}\mathbf{X}_t\right)^{-1} \qquad (10) \\ \overleftrightarrow{\boldsymbol{\mu}}_t(\boldsymbol{x},\overleftarrow{\boldsymbol{x}}) &= \boldsymbol{\Sigma}_t\left(\mathbf{Q}^{-1}\mathbf{F}\boldsymbol{x}+\mathbf{F}^\top\mathbf{Q}^{-1}\overleftarrow{\boldsymbol{x}}+\mathbf{X}_t^\top\mathbf{H}_t^{-1}(\boldsymbol{y}_t-\boldsymbol{h}_t)\right) \qquad (11)\end{aligned}$$

so

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

Alternatively, we can use a bootstrap like proposal distribution with

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

This is not optimal but faster.

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

Input:

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

Proposal distribution with density

$$q\left(\boldsymbol{\alpha}_t \middle| \boldsymbol{\alpha}_{t-1}^{(j)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \tag{12}$$

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 the density $p(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:t})$ 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 the artificial density $\widetilde{p}(\boldsymbol{\alpha}_t|\boldsymbol{y}_{t:d})$ for $t = d+1, d, d-1, \ldots, 1$. See Algorithm 3.

procedure SMOOTH (COMBINE)

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

Re-sample

$i = 1, 2, \ldots, N_s$ 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

$$\widehat{w}_t^{(i)} \propto \frac{f\left(\widehat{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}\right) g_t\left(\boldsymbol{y}_t \middle| \widehat{\boldsymbol{\alpha}}_t^{(i)}\right) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)} \middle| \widehat{\boldsymbol{\alpha}}_t^{(i)}\right) w_{t-1}^{(j_i)} \widetilde{w}_{t+1}^{(k_i)}}{\widetilde{q}\left(\widehat{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right) \beta_t^{(j_i)} \widetilde{\beta}_t^{(k_i)} \gamma_{t+1}\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k_i)}\right)} \tag{13}$$

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

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

1: Sample $\boldsymbol{\alpha}_0^{(1)}, \ldots, \boldsymbol{\alpha}_0^{(N_f)}$ particles from $\mathcal{N}(\boldsymbol{a}_0, \mathbf{Q}_0)$ 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 $\beta_t^{(j)}$ using h and re-sample according to $\beta_t^{(j)}$ to get indices $j_1, \ldots j_N$. If we do not re-sample then set $\beta_t^{(j)} = 1/N$ or $1/N_f$ at time $t = 1$.

5: **procedure** PROPAGATE

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

7: **procedure** RE-WEIGHT

8: Re-weight particles using

$$w_t^{(i)} \propto \frac{g_t\left(\boldsymbol{y}_t \middle| \boldsymbol{\alpha}_t^{(i)}\right) f\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}\right) w_{t-1}^{(j_i)}}{q\left(\boldsymbol{\alpha}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j_i)}, \boldsymbol{y}_t\right) \beta_t^{(j_i)}} \tag{14}$$

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

Input:

An artificial distribution

$$\widetilde{p}\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{t:d}\right) \propto \gamma_t\left(\boldsymbol{\alpha}_t\right) p\left(\boldsymbol{y}_{t:d}|\boldsymbol{\alpha}_t\right) \tag{15}$$

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

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

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

5: **procedure** PROPAGATE

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

7: **procedure** RE-WEIGHT

8: Re-weight particles using

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

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 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. 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,

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

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

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

Similar arguments does not work for the re-sampling weights in the forward particle filter

in Equation (2). However, we can use

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

as in Fearnhead et al. (2010). We can approximate the backwards particle filter re-sampling weights in equation (9) in a similar way

$$
\begin{aligned}
\widetilde{\beta}_t^{(k)} &\propto \widetilde{p}\left(\boldsymbol{y}_t \middle| \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \widetilde{w}_{t+1}^{(k)} \\
&\approx \frac{g_t\left(\boldsymbol{y}_t|\widehat{\boldsymbol{\alpha}}\right) \widetilde{p}\left(\widehat{\boldsymbol{\alpha}} \middle| \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \widetilde{w}_{t+1}^{(k)}}{\widetilde{q}\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right)} \\
&= \frac{g_t\left(\boldsymbol{y}_t|\widehat{\boldsymbol{\alpha}}\right) f\left(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)} \middle| \widehat{\boldsymbol{\alpha}}\right) \gamma_t(\widehat{\boldsymbol{\alpha}}) \widetilde{w}_{t+1}^{(k)}}{\widetilde{q}\left(\widehat{\boldsymbol{\alpha}} \middle| \boldsymbol{y}_t, \widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}\right) \gamma_{t+1}(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)})} \qquad (17) \\
\widehat{\boldsymbol{\alpha}} &= \overleftarrow{\boldsymbol{\mu}}(\widetilde{\boldsymbol{\alpha}}_{t+1}^{(k)}, \boldsymbol{z}) \\
\overleftarrow{\boldsymbol{\mu}}(\boldsymbol{x}, \boldsymbol{z}) &= \overleftarrow{\boldsymbol{\Sigma}}_t(\boldsymbol{z}) \left(\mathbf{P}_t^{-1}\boldsymbol{m}_t + \mathbf{F}^\top \mathbf{Q}^{-1}\boldsymbol{x} + \mathbf{X}_t^\top \mathbf{G}_t(\boldsymbol{z})\boldsymbol{u}_t(\boldsymbol{z})\right) \qquad (18) \\
\overleftarrow{\boldsymbol{\Sigma}}_t(\boldsymbol{z}) &= \left(\mathbf{P}_t^{-1} + \mathbf{F}^\top \mathbf{Q}^{-1}\mathbf{F} + \mathbf{X}_t^\top \mathbf{G}_t(\boldsymbol{z})\mathbf{X}_t\right)^{-1} \qquad (19)
\end{aligned}
$$

We may also use a multivariate t-distribution for the proposal distribution to get heavier tails then we do with the multivariate normal distribution.

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 $\mathcal{O}\left(p^2 n_{\max} N_S\right)$ or $\mathcal{O}\left(p^2 n_{\max} N\right)$ 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 $\mathcal{O}\left(p n_{\max} N_S + p^2 n_{\max}\right)$ or $\mathcal{O}\left(p n_{\max} N + p^2 n_{\max}\right)$.

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 auto-regression, 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

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

where δ is the Dirac delta function and the second equality follows in the particular example. 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)

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

where the last equality holds in this example. This distribution is obviously degenerate as it only has mass at the point $\boldsymbol{\xi}_t = \mathbf{K}\boldsymbol{\alpha}_{t+1}$.

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

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

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,

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

with which we can find that the joint distribution is

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

We are now able to compute

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

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

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

which replaces the mean and covariance matrix in Equation (18) and (19), $\boldsymbol{z}_t \in \mathbb{R}^r$ 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 $\boldsymbol{\xi}_t$. The new proposal distribution is used in Equation (17) when we compute the re-sampling weights.

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 (11) and (10) with the approximation

$$\overleftrightarrow{\boldsymbol{\Sigma}}_t(\boldsymbol{z}) = \left(\mathbf{Q}^{-1} + \mathbf{V}_t^\top \boldsymbol{\Sigma}^{-1}_{\alpha_{t+o}|\boldsymbol{\xi}_t} \mathbf{V}_t + \mathbf{X}_t^\top \mathbf{G}_t(\boldsymbol{z}) \mathbf{X}_t\right)^{-1}$$

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

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

4 Log likelihood evaluation

We can evaluate 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, page 5) and Malik and Pitt (2011, page 193) using the forward particle filter shown in Algorithm 2.

5 Parameter inference

In this section I first show an example of parameter estimation the first order random walk using EM-algorithm (Dempster et al., 1977). Then I cover the general vector auto-regression model and how one can estimate the fixed effects. Lastly, I will turn to estimation of observed information matrix.

The formulas for parameter estimation for the first order random with the 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 $\boldsymbol{\varphi}$. Then we compute

$$\begin{aligned} \boldsymbol{t}_t^{(\boldsymbol{\varphi})} &\approx \sum_{i=1}^{N_s} \widehat{\boldsymbol{\alpha}}_t^{(i)} \widehat{w}_t^{(i)} \\ \boldsymbol{T}_t^{(\boldsymbol{\varphi})} &\approx \sum_{i=1}^{N_s} \left(\widehat{\boldsymbol{\alpha}}_t^{(i)} - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j_{it})}\right)\left(\widehat{\boldsymbol{\alpha}}_t^{(i)} - \mathbf{F}\boldsymbol{\alpha}_{t-1}^{(j_{it})}\right)^\top \widehat{w}_t^{(i)} \end{aligned} \tag{20}$$

where $\boldsymbol{\alpha}_{s:t} = \{\boldsymbol{\alpha}_s, \boldsymbol{\alpha}_{s+1}, \dots \boldsymbol{\alpha}_t\}$, 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 i'th smoothed particle at time t The update of $\boldsymbol{\mu}_0$ and $\mathbf{Q}$ given the summary statistics is

$$\boldsymbol{\mu}_0 = \boldsymbol{t}_0^{(\boldsymbol{\varphi})} \qquad \mathbf{Q} = \frac{1}{d-1} \sum_{t=2}^{d} \mathbf{R}^{+} \boldsymbol{T}_t^{(\boldsymbol{\varphi})} \mathbf{R}^{+\top} \tag{21}$$

We then take another iteration of the EM algorithm with the new $\boldsymbol{\mu}_0$ and $\mathbf{Q}$ 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.

I do not think we can estimate $\boldsymbol{a}_0$ consistently so we likely just want to fix it at some value...

5.1 Vector auto-regression models

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

$$\begin{aligned}
\mathbf{N} &= \left(\widehat{\boldsymbol{\alpha}}_2^{(1)}, \widehat{\boldsymbol{\alpha}}_2^{(2)}, \dots, \widehat{\boldsymbol{\alpha}}_2^{(N_s)}, \widehat{\boldsymbol{\alpha}}_3^{(1)}, \dots, \widehat{\boldsymbol{\alpha}}_d^{(N_s)}\right)^\top \mathbf{R}^{+\top} \\
\mathbf{M} &= \left(\boldsymbol{\alpha}_1^{(j_{12})}, \boldsymbol{\alpha}_1^{(j_{22})}, \dots, \boldsymbol{\alpha}_1^{(j_{N_s 2})}, \boldsymbol{\alpha}_2^{(j_{13})}, \dots, \boldsymbol{\alpha}_{d-1}^{(j_{N_s d})}\right)^\top \\
\mathbf{W} &= \operatorname{diag}\left(\widehat{w}_2^{(1)}, \dots, \widehat{w}_2^{(N_s)}, \widehat{w}_3^{(1)}, \dots, \widehat{w}_d^{(N_s)}\right)
\end{aligned}$$

where $\operatorname{diag}(\cdot)$ is a diagonal matrix which diagonal elements are the argument. We implicitly let the above depend 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 show that the M-step maximizers are

$$\widehat{\mathbf{F}}^\top \mathbf{R}^{+\top} = \left(\mathbf{M}^\top \mathbf{W} \mathbf{M}\right)^{-1} \mathbf{M}^\top \mathbf{W} \mathbf{N} \tag{22}$$

$$\widehat{\mathbf{Q}} = \frac{1}{d-1} \left(\mathbf{N} - \mathbf{R}^+ \widehat{\mathbf{F}} \mathbf{M}\right)^\top \mathbf{W} \left(\mathbf{N} - \mathbf{R}^+ \widehat{\mathbf{F}} \mathbf{M}\right) \tag{23}$$

which is the typical vector auto-regression estimators with weights. Equation (22) and (23) can easily be computed in parallel using QR decompositions as in the `bam` 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 computation here have a computation time which is independent of the number of observations. Thus, the computation involved here is often fast as the dimension of the state vector is small.

5.2 Restricted vector auto-regression models

Suppose that we want to restrict some of the parameters of $\mathbf{F}$ and $\mathbf{Q}$. E.g., we can restrict the model to

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

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

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

with

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

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

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

$\mathbf{G} \in \mathbb{R}^{rp \times g}$ 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 $\boldsymbol{\theta}$. Similarly, 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, on the, non-Monte Carlo, expectation maximization algorithm). The first conditional maximization step is

$$\boldsymbol{\theta}^{(i+1)} = \mathbf{G}^+ \left(\mathbf{Q}^{(i)} \otimes \left(\mathbf{M}^\top \mathbf{W} \mathbf{M}\right)^{-1}\right) \mathbf{G}^{+\top} \mathbf{G}^\top \text{vec}\left(\mathbf{M}^\top \mathbf{W} \mathbf{N} \mathbf{Q}^{-(i)}\right) \tag{24}$$

where $\otimes$ is the Kronecker product and $\mathbf{Q}^{-(i)}$ is the inverse of $\mathbf{Q}^{(i)}$. Equation (24) is easily computed with the QR decomposition we make to compute for Equation (22). Having obtained the new $\boldsymbol{\theta}^{(i+1)}$, we update the variable elements of $\mathbf{F}$ (those columns "picked out" by $\mathbf{R}$ in $\mathbf{R}^+\mathbf{F}$) and denote the new estimate $\widehat{\mathbf{F}}^{(i+1)}$. The second conditional maximization step which updates $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ is

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

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. $\mathbf{C}$ will not be a valid correlation matrix for all $\boldsymbol{\phi} \in \mathbb{R}^k$ for some choices of $\mathbf{K}$. The invalid values are ruled out doing the numerical optimization. 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 $n_{\max}$. 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.

5.3 Estimating fixed effect coefficients

Next, we turn to estimating the fixed effects 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 5.1. 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...

5.4 Observed information matrix

Computing the observed information matrix requires an application of the missing information principle (Louis, 1982). However we cannot evaluate the quantities we need with the output from Algorithm 1 since we only have discrete approximation of the smoothed distribution of triplet of particles, $\boldsymbol{\alpha}_{t-1:t+1}$, but need an approximation for the entire path, $\boldsymbol{\alpha}_{1:d}$. One solution is to use the methods covered in e.g., Cappé et al. (2005, section 8.3 and chapter 11) and Poyiadjis et al. (2011). The method in Cappé et al. (2005) only requires the forward particle filter output. It is though not implemented.

6 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 the 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 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. (2009) 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 $\mathcal{O}\left(N \log(N)\right)$ run times 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 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 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 the parameters $\boldsymbol{\theta}$ at each time t and use the methods in Andrieu and Doucet (2002) or similar ideas to perform online estimation.

7 Briers et al. (2009)

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 (27) comes from the generalized two-filter formula. To motivate the smoother, we use the following result

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

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

$$\begin{aligned}
p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:d}\right) &= \frac{p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:t-1}\right)p\left(\boldsymbol{y}_{t:d}|\boldsymbol{\alpha}_t\right)}{p\left(\boldsymbol{y}_{t:d}|\boldsymbol{y}_{1:t-1}\right)} \\
&\propto p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:t-1}\right)p\left(\boldsymbol{y}_{t:d}|\boldsymbol{\alpha}_t\right) \\
&= p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:t-1}\right)\widetilde{p}\left(\boldsymbol{y}_{t:d}\right)\frac{\widetilde{p}\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{t:d}\right)}{\gamma_t(\boldsymbol{\alpha}_t)} \\
&\propto p\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{1:t-1}\right)\frac{\widetilde{p}\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{t:d}\right)}{\gamma_t(\boldsymbol{\alpha}_t)} \\
&= \widetilde{p}\left(\boldsymbol{\alpha}_t|\boldsymbol{y}_{t:d}\right)\frac{\left[\int p\left(\boldsymbol{\alpha}_{t-1}|\boldsymbol{y}_{1:t-1}\right)f\left(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}\right)\partial\boldsymbol{\alpha}_{t-1}\right]}{\gamma_t(\boldsymbol{\alpha}_t)} \\
&\overset{\propto}{\sim} \sum_{i=1}^{N}\widetilde{w}_t^{(i)}\delta_{\widetilde{\boldsymbol{\alpha}}_t^{(i)}}\left(\boldsymbol{\alpha}_t\right)\frac{\left[\sum_{j=1}^{N}w_{t-1}^{(j)}f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\middle|\boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]}{\gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)}
\end{aligned} \tag{25}$$

Similar arguments leads to

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

where

$$\widehat{w}_t^{(i,j)} = \widehat{w}_t^{(i)}\frac{w_{t-1}^{(j)}f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\middle|\boldsymbol{\alpha}_{t-1}^{(j)}\right)}{\left[\sum_{j=1}^{N}w_{t-1}^{(j)}f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\middle|\boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]} \tag{26}$$

We need the latter for the EM-algorithm.

Algorithm 4 $\mathcal{O}\left(N^2\right)$ generalized two-filter smoother using the method in 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

$$\widehat{w}_t^{(i)} \propto \widetilde{w}_t^{(i)} \frac{\left[\sum_{j=1}^N w_{t-1}^{(j)} f\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)} \middle| \boldsymbol{\alpha}_{t-1}^{(j)}\right)\right]}{\gamma_t\left(\widetilde{\boldsymbol{\alpha}}_t^{(i)}\right)} \tag{27}$$

With the result above, we can show the reasoning behind cover the smoother from Fearnhead et al. (2010). Similar to Equation (25) we find that

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

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, $\widetilde{\boldsymbol{\alpha}}_{t+1}^{(i)}$, 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

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

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

8 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 particles 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 $d + 1$.

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.
- `"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 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 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 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.

8.1 Linear maps

The methods describe above involves many linear maps. These are implemented with `C++` abstract classes with specialized `map` member functions for particular problem to decrease

the computation. An alternative would have been to use a sparse matrix implementation. It does not matter much in terms of computation time if n_t is much larger then the state vector's dimension. However, this section is included if anyone ever looks through the code. As 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 == left`, and $\mathbf{X}\mathbf{B}^\top$ if `s == right`.

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`	$\mathbf{F}^{-1}$

Further, we will need function to compute terms

$$\overleftarrow{\mathbf{P}}_t\mathbf{F}^\top\overleftarrow{\mathbf{P}}_{t+1}^{-1}\boldsymbol{\alpha}_{t+1} + \overleftarrow{\mathbf{S}}_t\overleftarrow{\mathbf{P}}_t^{-1}\overleftarrow{\boldsymbol{m}}_t$$

for an arbitrary $\boldsymbol{\alpha}_{t+1}$, $\overleftarrow{\mathbf{S}}_t = \overleftarrow{\mathbf{P}}_t\mathbf{F}^\top\overleftarrow{\mathbf{P}}_{t+1}^{-1}\mathbf{R}\mathbf{Q}\mathbf{R}^\top\mathbf{F}^{-\top}$, $\overleftarrow{\mathbf{P}}_t^{-1}\overleftarrow{\boldsymbol{m}}_t$, and $\overleftarrow{\mathbf{P}}_t^{-1}$. This is done with the methods `bw_mean(signed int, const arma::vec&)`, `bw_covar(signed int)`, `uncond_mean_term(signed int)`, and `uncond_covar_inv(signed int)`. The terms and factors that can will computed once using Equation (8) are computed and stored.

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, 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.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, Jul 2000. ISSN 1573-1375. doi: 10.1023/A:1008935410038. URL `https://doi.org/10.1023/A:1008935410038`.

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.

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.