A DIVIDE-AND-CONQUER SEQUENTIAL MONTE CARLO APPROACH TO HIGH DIMENSIONAL FILTERING

Department of Statistics, University of Warwick

Supplementary Material

Supplementary materials contains details of the tempering approach, additional details and results on the experiments.

S1 Comparison of Resampling Strategies

The divide and conquer SMC algorithm described in Section 2.3 merges the particle populations of each node's children using mixture resampling. We compare the performances of lightweight mixture resampling with its adaptive version described in Section 3.3, those of full-cost mixture resampling, in which all possible N^2 permutations of the particles are created, and those of a linear cost version of DaC-SMC in which no mixing weights are used (Lindsten et al., 2017).

To compare the four resampling schemes we consider the mean squared

error (MSE) for each component i using the example of Section 4.1,

$$\mathrm{MSE}(x_t(i)) := \mathbb{E}\left[(\bar{x}_t(i) - \mu_{t,i})^2\right], \tag{S1.1}$$

where $\bar{x}_t(i)$ denotes the estimate of the mean of component i at time t and $\mu_{t,i}$ denotes the true mean of $x_t(i)|y_{1:t}$ obtained from the Kalman filter. To approximate (S1.1) we consider an empirical average over 50 repetitions.

Figure 1 shows the average (over component) MSE of the estimates obtained for $d = 128$ and $t = 10$ time steps as a function of runtime. The linear cost version of DaC-SMC has the smallest runtime, however, the results in terms of MSE are the worst among the four algorithms. In fact, this linear cost version does not use mixture weights at the point of selection, and therefore does not take into account the mismatch between $\gamma_{t,\mathcal{C}_u}$ and $\gamma_{t,u}$ when resampling (this is corrected for with a subsequent importance reweighting). The full mixture resampling has a higher cost, and becomes unmanageable for large N (it is, in fact, not included for $N = 1000, 5000$); the non-adaptive version of lightweight mixture resampling also becomes too expensive for large N although the increase in cost is less steep than that of the full mixture resampling (the non-adaptive lightweight mixture resampling has manageable cost for $N = 1000$ but it is not included for $N = 5000$). The MSE achieved by the non-adaptive lightweight mixture

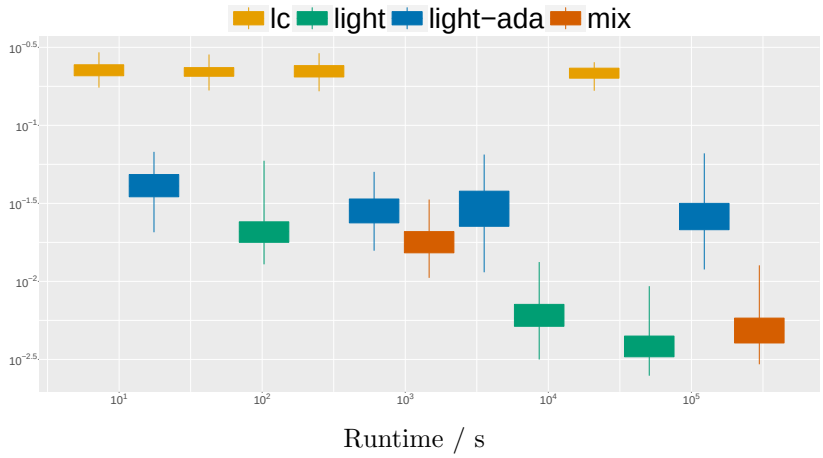

Figure 1: Average (over dimensions) MSE at $t = 10$ over 50 runs for $d = 128$ as function of runtime for the linear cost version of DaC in Lindsten et al. (2017) and Algorithm 2 with lightweight mixture resampling, adaptive lightweight mixture resampling and full mixture resampling. The boxes, from left to right, correspond to increasing number of particles, $N = 100, 500, 1000, 5000$. Due to the excessive cost, the results for mixture resampling are not reported for $N = 1000, 5000$ and those for lightweight mixture resampling with no adaptation are not reported for $N = 5000$.

resampling is equivalent to that of the full mixture resampling, at a considerably lower cost. For small N, the adaptive lightweight mixture resampling also gives comparable results, but, as N increases, the MSE stops improving and eventually settles around 0.02. This is likely due to the fact that, as it is the case of Figure 2, sometimes the target ESS is not reached at level 1 and more permutations would be needed to obtain a good importance sampling proposal. One could then consider to add tempering steps to obtain better samples or, alternatively, to increase the target ESS. By how much the target ESS should be increased is a challenging question which we have not investigated further in this work.

S2 Tempering Strategy

If the product of the marginals over the two child nodes merged via mixture resampling provides a poor approximation for $\gamma_{t,u}$ (which in some circumstances could be detected, for example, by Algorithm 3 failing to achieve the target ESS), one could expect to alleviate this mismatch via a tempering strategy. Here, we describe an adaptive tempering strategy of the sort described in Lindsten et al. (2017, Section 4.2) adapted to our context.

We define the following sequence of targets $\hat{\pi}_{\alpha_{u,j}} = \hat{\gamma}_{\alpha_{u,j}}/\hat{\gamma}_{\alpha_{u,j}}(1)$ where

$$\hat{\gamma}_{\alpha_{u,j}}(z_{t,u,j}) = [\gamma_{t,\mathcal{C}_u}(z_{t,u,j})]^{1-\alpha_{u,j}}\, \gamma_{t,u}(z_{t,u,j})^{\alpha_{u,j}}$$

for $z_{t,u,j} \in \mathbb{R}^{|\mathcal{V}_u|}$ and $\alpha_{u,j} \in [0,1]$, which interpolates from the proposal $\gamma_{t,\mathcal{C}_u}$ to the target at node u, $\gamma_{t,u}$. In a standard approach to tempering we would start from $\alpha_{u,j} = 0$ and select the next value adaptively (see e.g. Wang et al. (2020); Zhou et al. (2016)). However, Algorithm 3 corresponds to a first tempering step which moves the proposal $\gamma_{t,\mathcal{C}_u}$ closer to the target $\gamma_{t,u}$; therefore, instead of starting the tempering schedule at $\alpha_{u,j} = 0$, we identify the value of $\alpha^\star$ which corresponds to the intermediate tempered target obtained after mixture resampling and carry on with tempering from

there until $\alpha_{u,j} = 1$. To identify the value of α we consider the ESS

$$\mathrm{ESS}(\alpha) := \left(\sum_{n=1}^{N} (\tilde{w}_{t,u}^n)^\alpha\right)^2 / \sum_{n=1}^{N} (\tilde{w}_{t,u}^n)^{2\alpha},$$

and solve $\mathrm{ESS}(\alpha) = \mathrm{ESS}^\star$, where $\mathrm{ESS}^\star$ is the target ESS in Section 3.3. The equally weighted particles after mixture resampling, $\{z_{t,\mathcal{C}_u}, \omega_{u,j} = 1\}_{n=1}^N$, approximate $\hat{\pi}_{\alpha^\star}$. The following values of the tempering sequence are chosen adaptively as described in, e.g., Zhou et al. (2016, Section 3.3.2 and Algorithm 4) and Wang et al. (2020, Algorithm 4), using the conditional ESS

$$\mathrm{CESS}_j(\alpha) := N \left(\sum_{n=1}^{N} \bar{\omega}_{u,j}^n (\tilde{w}_{t,u}^n)^{\alpha - \alpha_{u,j-1}}\right)^2 / \sum_{n=1}^{N} \bar{\omega}_{u,j}^n (\tilde{w}_{t,u}^n)^{2(\alpha - \alpha_{u,j-1})}, \quad \text{(S2.2)}$$

where $\bar{\omega}_{u,j}^n \propto \omega_{u,j}^n$. Given a decay threshold $\beta > 0$, we find the next value $\alpha_{u,j}$ solving $\mathrm{CESS}_j(\alpha) = \beta$, then we update the weights, perform a resampling step if necessary, and rejuvenate the particles using a $\hat{\pi}_{\alpha_{u,j}}$-invariant kernel $K'_{\alpha_{u,j}}$. This process is repeated until $\alpha_{u,j} = 1$. The resulting tempering strategy is summarized in Algorithm 1.

S3 Further Details on the Experiments

We collect here further details on the models considered in Section 4.

Algorithm 1 Tempering.

Initialize: set $\alpha_{u,1}$ to be the solution of $\mathrm{ESS}(\alpha) = \mathrm{ESS}^\star$ and set $\omega_{u,1}^n = 1$ for $n \leq N$.

Set: $j = 1$.
while $\alpha_{u,j} < 1$ **do**
  Adapt: set $j = j + 1$ and find $\alpha_{u,j}$ solving $\mathrm{CESS}_j(\alpha) = \beta$ for $\alpha \in (\alpha_{u,j-1}, 1)$ using bisection.
  Reweight: compute the weights $\omega_{u,j}^n = \omega_{u,j-1}^{n,N} (\tilde{w}_{t,u}^n)^{\alpha_{u,j} - \alpha_{u,j-1}}$ for $n \leq N$ and compute the ESS.
  if $\mathrm{ESS} < N/2$ **then**
    Resample: draw new $z_{t,u,j-1}^{n,N}$ independently with weights $\omega_{u,j}^n$ and update $\omega_{u,j}^n = 1$ for $n \leq N$.
  end if
  Propose: draw $z_{t,u,j}^n \sim K'_{\alpha_{u,j}}(\cdot, z_{t,u,j-1}^n)$ from a $\hat{\pi}_{\alpha_{u,j}}$ invariant kernel for $n \leq N$.
end while

S3.1 Simple Linear Gaussian Model

We consider the simple linear Gaussian model used in the experiments in Næsseth et al. (2015, Section 6.1):

$$
\begin{aligned}
X_1 &\sim \mathcal{N}_d(0, \mathrm{Id}_d) \\
X_t &= 0.5 A X_{t-1} + U_t, & U_t &\sim \mathcal{N}_d(0, \Sigma) \\
Y_t &= X_t + V_t, & V_t &\sim \mathcal{N}_d(0, \sigma_y^2 \mathrm{Id}_d),
\end{aligned}
$$

with $A = A_1 A_2^{-1}$ where

$$A_1 = \begin{pmatrix} \tau+\lambda & 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & \tau & 0 & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \tau & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \tau \end{pmatrix},$$

$$A_2 = \begin{pmatrix} \tau+\lambda & 0 & 0 & \dots & \dots & 0 & 0 \\ -\lambda & \tau+\lambda & 0 & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & -\lambda & \tau+\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & -\lambda & \tau+\lambda \end{pmatrix}^{-1},$$

and

$$\Sigma^{-1} = A_2^T \begin{pmatrix} \tau & 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & \tau+\lambda & 0 & 0 & \dots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \tau+\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \tau+\lambda \end{pmatrix} A_2,$$

with $\tau = \lambda = 1$ and $\sigma_y^2 = 0.5^2$.

To derive the mixture weights for our divide-and-conquer approach we write the joint density of $(X_{1:t}, Y_{1:t})$:

$$p(x_{1:t}, y_{1:t}) \propto \prod_{k=1}^{t} \left[\exp\left(-\frac{1}{2}(x_k - 0.5Ax_{k-1})^T \Sigma^{-1} (x_k - 0.5Ax_{k-1}) \right) \times \prod_{i=1}^{d} \mathcal{N}(y_k(i); x_k(i), \sigma_y^2) \right].$$

The functions $f_{t,u}, g_{t,u}$ in the definition of $\gamma_{t,u}$ in (3.6) are

$$f_{t,u}(x_{t-1}, x_t(\mathcal{V}_u)) \propto$$
$$\exp\left(-\frac{1}{2}(x_t(\mathcal{V}_u) - 0.5A_{|\mathcal{V}_u}x_{t-1}(\mathcal{V}_u))^T\Sigma^{-1}_{|\mathcal{V}_u}(x_t(\mathcal{V}_u) - 0.5A_{|\mathcal{V}_u}x_{t-1}(\mathcal{V}_u))\right),$$
$$g_{t,u}(x_t(\mathcal{V}_u), (y_t(i))_{i\in\mathcal{V}_u}) \propto \prod_{i\in\mathcal{V}_u}\mathcal{N}(y_t(i); x_t(i), \sigma_y^2),$$

where we denote $x_t(\mathcal{V}_u) = (x_t(i))_{i\in\mathcal{V}_u}$ and $A_{|\mathcal{V}_u}, \Sigma^{-1}_{|\mathcal{V}_u}$ are the restrictions of A, Σ^{-1}, respectively, to $\mathcal{V}_u$, i.e. $A_{|\mathcal{V}_u}$ is the matrix obtained discarding the elements of A corresponding to components which are not in $\mathcal{V}_u$, and similarly for $\Sigma^{-1}_{|\mathcal{V}_u}$.

Expanding the quadratic form in the display above, we find that we can decouple the dependence on the current time step from that on the past and decompose

$$f_{t,u}(x_{t-1}, (x_t(i))_{i\in\mathcal{V}_u}) = f^{(1)}_{t,u}(x_{t-1}, (x_t(i))_{i\in\mathcal{V}_u}) \prod_{i\in\mathcal{V}_u, i\neq j^1_u}\tilde{f}(x_t(i-1), x_t(i)),$$

with

$$f_{t,u}^{(1)}(x_{t-1},(x_t(i))_{i\in\mathcal{V}_u}) \propto \prod_{i\in\mathcal{V}_u, i\neq j_u^1}\left[\exp\left(-\frac{\tau+\lambda}{2}\left(x_t(i)-0.5\frac{\tau}{\tau+\lambda}x_{t-1}(i)\right)^2\right)\right.$$
$$\left.\times\exp\left(-\frac{1}{2}\frac{\lambda\tau}{\tau+\lambda}x_t(i-1)x_{t-1}(i)\right)\right]$$
$$\times\begin{cases}\exp\left(-\frac{\tau+\lambda}{2}\left(x_t(j_u^1)-0.5\frac{\tau}{\tau+\lambda}x_{t-1}(j_u^1)\right)^2\right) & \text{if } j_u^1\neq 1\\ \exp\left(-\frac{\tau}{2}(x_t(1)-0.5x_{t-1}(1))^2\right) & \text{if } j_u^1=1\end{cases},$$

where j_u^1 denotes the first index associated with node u, and

$$\tilde{f}(s_1,s_2) := \exp\left(-\frac{1}{2}\left[\frac{\lambda^2}{\tau+\lambda}s_1^2-2\lambda s_1 s_2\right]\right).$$

The mixture weights are then given by

$$m_{t,u}(z_{t,\mathcal{C}_u}) \propto \tilde{f}\left(z_{t,\ell(u)}(i_{\ell(u)}^{n_{\ell(u)}}), z_{t,r(u)}(i_{r(u)}^1)\right)$$
$$\times\frac{\sum_{n=1}^N f_{t,u}^{(1)}(z_{t-1,\mathfrak{R}}^n, z_{t,\mathcal{C}_u})}{\sum_{n=1}^N f_{t,\ell(u)}^{(1)}(z_{t-1,\mathfrak{R}}^n, z_{t,\ell(u)})\sum_{n=1}^N f_{t,r(u)}^{(1)}(z_{t-1,\mathfrak{R}}^n, z_{t,r(u)})},$$

where we recall that $z_{t,u}$ in (3.6) corresponds to $z_{t,u} = (x_t(i))_{i\in\mathcal{V}_u}$, $i_{\ell(u)}^{n_{\ell(u)}}$ is the last index associated with node $\ell(u)$ and $i_{r(u)}^1$ is the first index associated with $r(u)$.

S3.2 Spatial Model

The joint density of $(X_{1:t}, Y_{1:t})$ is

$$p(x_{1:t}, y_{1:t}) \propto \prod_{k=1}^{t} \left(\prod_{v \in V} \mathcal{N}\left(x_k(v); x_{k-1}(v), \sigma_x^2\right) \times \left[1 + \nu^{-1} \sum_{v \in V} \left((y_k(v) - x_k(v)) \sum_{j: D(v,j) \le r_y} \tau^{D(v,j)} (y_k(j) - x_k(j)) \right) \right]^{-(\nu + |V|)/2} \right)$$

with initial distribution $X_1 \sim \prod_{v \in V} \mathcal{N}\left(x_k(v); 0, \sigma_x^2\right)$. The functions $f_{t,u}, g_{t,u}$ in the definition of $\gamma_{t,u}$ in (3.6) are

$$f_{t,u}(x_{t-1}, (x_t(i))_{i \in \mathcal{V}_u}) \propto \prod_{v \in \mathcal{V}_u} \mathcal{N}\left(x_k(v); x_{k-1}(v), \sigma_x^2\right)$$

$$g_{t,u}((x_t(i))_{i \in \mathcal{V}_u}, (y_t(i))_{i \in \mathcal{V}_u}) \propto \left[1 + \nu^{-1} \sum_{v \in \mathcal{V}_u} \left((y_k(v) - x_k(v)) \sum_{j: D(v,j) \le r_y, j \in \mathcal{V}_u} \tau^{D(v,j)} (y_k(j) - x_k(j)) \right) \right]^{-(\nu + |\mathcal{V}_u|)/2},$$

where we recall that $z_{t,u}$ in (3.6) corresponds to $z_{t,u} = (x_t(i))_{i \in \mathcal{V}_u}$. From the above, we obtain that the mixture weights (3.10) are given by $m_{t,u}(z_{t,\mathcal{C}_u}) = R^f_{t,u}(z_{t,\mathcal{C}_u}) R^g_{t,u}(z_{t,\mathcal{C}_u})$, where

$$\begin{aligned} R^f_{t,u}(z_{t,\mathcal{C}_u}) &= \frac{N^{-1} \sum_{n=1}^{N} f_{t,u}(z^n_{t-1,\mathfrak{R}}, z_{t,\mathcal{C}_u})}{N^{-1} \sum_{n=1}^{N} f_{t,\ell(u)}(z^n_{t-1,\mathfrak{R}}, z_{t,\ell(u)}) N^{-1} \sum_{n=1}^{N} f_{t,r(u)}(z^n_{t-1,\mathfrak{R}}, z_{t,r(u)})} \\ &\propto \frac{\sum_{n=1}^{N} \prod_{v \in \mathcal{V}_u} \mathcal{N}\left(z_{t,\mathcal{C}_u}(v); z^n_{t-1,\mathfrak{R}}(v), \sigma_x^2\right)}{\sum_{n_1=1}^{N} \prod_{v \in \mathcal{V}_{\ell(u)}} \mathcal{N}\left(z_{t,\ell(u)}(v); z^{n_1}_{t-1,\mathfrak{R}}(v), \sigma_x^2\right) \sum_{n_2=1}^{N} \prod_{v \in r(u)} \mathcal{N}\left(z_{t,r(u)}(v); z^{n_2}_{t-1,\mathfrak{R}}(v), \sigma_x^2\right)} \end{aligned}$$

and

$$R^g_{t,u}(z_{t,\mathcal{C}_u}) = \frac{g_{t,u}(z_{t,\mathcal{C}_u}, (y_t(i))_{i\in\mathcal{V}_u})}{g_{t,\ell(u)}(z_{t,\ell(u)}, (y_t(i))_{i\in\mathcal{V}_{\ell(u)}}) g_{t,r(u)}(z_{t,r(u)}, (y_t(i))_{i\in\mathcal{V}_{r(u)}})}$$

$$\propto \left[1 + \nu^{-1} \sum_{v\in\mathcal{V}_u} \left((y_t(v) - z_{t,\mathcal{C}_u}(v)) \sum_{\substack{j: D(v,j)\leq r_y \\ j\in\mathcal{V}_u}} \tau^{D(v,j)} (y_t(j) - z_{t,\mathcal{C}_u}(j)) \right)\right]^{-\frac{\nu+|\mathcal{V}_u|}{2}}$$

$$\times \left[1 + \nu^{-1} \sum_{v\in\mathcal{V}_{\ell(u)}} \left((y_t(v) - z_{t,\ell(u)}(v)) \sum_{\substack{j: D(v,j)\leq r_y \\ j\in\mathcal{V}_{\ell(u)}}} \tau^{D(v,j)} (y_t(j) - z_{t,\ell(u)}(j)) \right)\right]^{\frac{\nu+|\mathcal{V}_{\ell(u)}|}{2}}$$

$$\times \left[1 + \nu^{-1} \sum_{v\in\mathcal{V}_{r(u)}} \left((y_t(v) - z_{t,r(u)}(v)) \sum_{\substack{j: D(v,j)\leq r_y \\ j\in\mathcal{V}_{r(u)}}} \tau^{D(v,j)} (y_t(j) - z_{t,r(u)}(j)) \right)\right]^{\frac{\nu+|\mathcal{V}_{r(u)}|}{2}}.$$

S4 Additional Results for the Experiments

S4.1 Simple Linear Gaussian Model

To further characterize the quality of the estimates obtained with Algorithm 2, we consider the relative mean squared error (RMSE) for component i at time t,

$$\mathrm{RMSE}(x_t(i)) := \frac{\mathbb{E}\left[(\bar{x}_t(i) - \mu_{t,i})^2\right]}{\sigma^2_{t,i}},$$

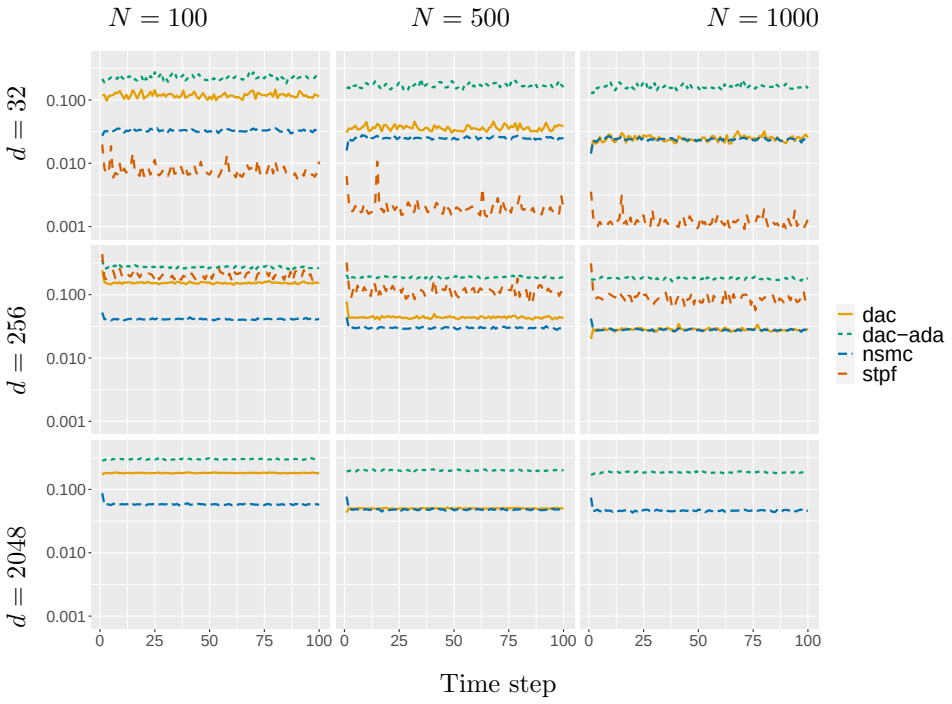

Figure 2: Average (over dimension) RMSE for 50 runs of DaC, NSMC and STPF. Due to their excessive cost, we do not include the results for STPF with $d = 2048$ and those of the non-adaptive version of DaC for $d = 2048, N = 1000$.

where $\bar{x}_t(i)$ denotes the estimate of the mean of component i at time t and $\mu_{t,i}, \sigma^2_{t,i}$ denote the true mean and true variance of $x_t(i)|y_{1:t}$ obtained from the Kalman filter. As for (S1.1), we approximate the RMSE using an empirical average over 50 repetitions.

As observed in Section 4.1, for small d, STPF performs better than DaC and NSMC (Figure 2 top row), however, as the dimension increases, the estimates provided by STPF become poor and their computational cost is prohibitive for large d (Figure 2 bottom row). Contrary to the other

approaches, NSMC does not seem to significantly improve as N increases.

S4.2 Spatial Model

To validate the results for the spatial model in Section 4.2 we compare the approximations obtained by Algorithm 2 with both adaptive and non-adaptive lightweight mixture resampling with those of the standard bootstrap particle filter (see, e.g., Chopin and Papaspiliopoulos (2020, Chapter 10)) on a 2×2 grid (Figure 3). Given the low dimensionality of the state space, we expect the bootstrap PF with a large number of particles to provide a good proxy for the filtering distribution $p(x_t|y_{1:t})$. The results for the three algorithms seem to be in good agreement, and the variance of both DaC algorithms with large $N = 5 \cdot 10^3, 10^4$ is comparable with that of the particle filter with $N = 10^5$.

To show how the increasing dimension causes instability of the simple bootstrap PF we report in Figure 4 the filtering mean estimates for two nodes of a 4×4 grid. Contrary to Figure 3, in which the bootstrap PF with $N = 10^5$ particles provides accurate estimates (first quartile, mean and third quartile coincide), when the dimension slightly increases the variance of the particle filter with $N = 10^5$ blows up and the recovered estimates are poor. On the other hand, both DaC approaches are stable and provide

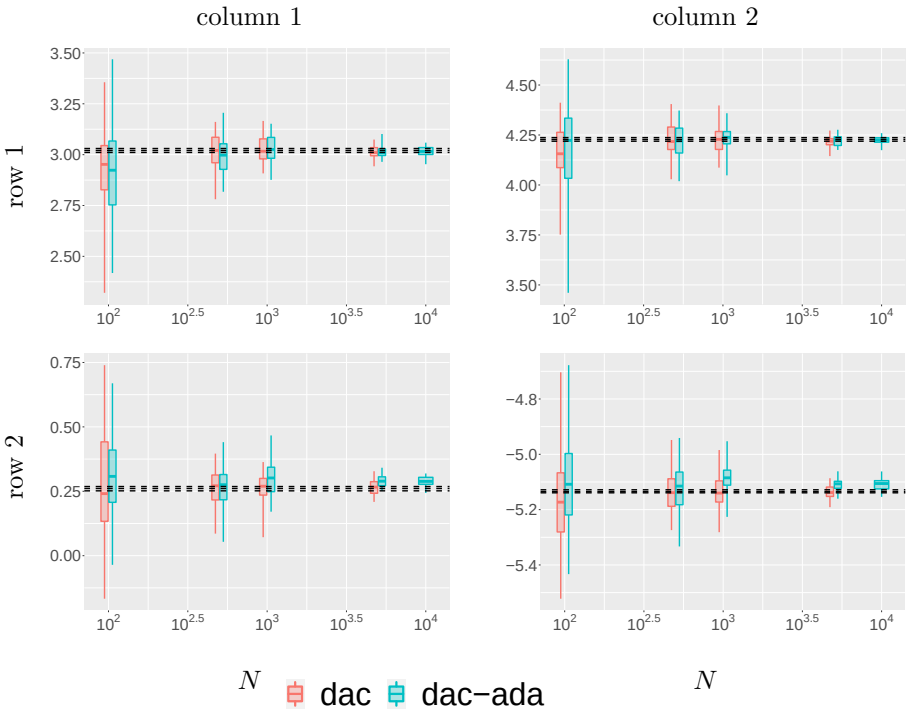

Figure 3: Filtering mean estimates obtained with DaC on a 2×2 grid. The reference lines show the average value of the filtering mean estimate and the interquartile range obtained with 50 repetitions of a bootstrap PF with $N = 10^5$ particles. The boxplots from left to right report the distributions over 50 repetitions for $N = 100, 500, 1000, 5000$ and 10000. The results for the non-adaptive version of DaC and $N = 10000$ are not included due to the excessive cost.

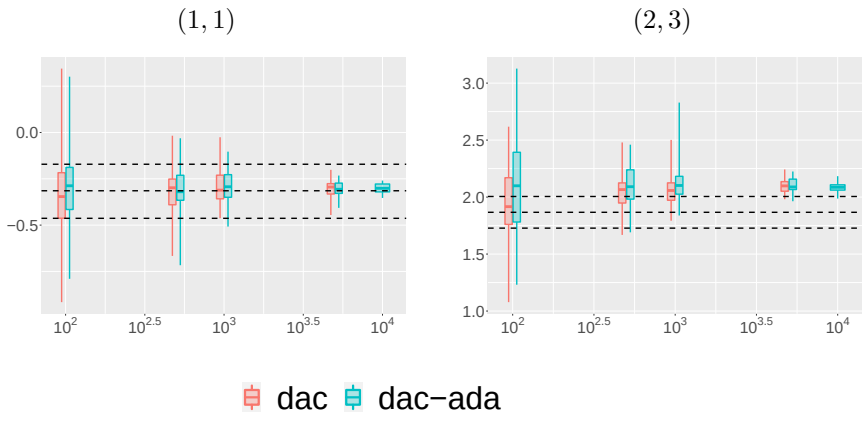

Figure 4: Filtering mean estimates obtained with DaC for node $(1,1)$ and $(2,3)$ of a 4×4 grid. The reference lines show the average value of the filtering mean estimate and the interquartile range obtained with 50 repetitions of a bootstrap PF with $N = 10^5$ particles. The boxplots from left to right report the distributions over 50 repetitions for $N = 100, 500, 1000, 5000$ and 10000. The results for the non-adaptive version of DaC and $N = 10000$ are not included due to the excessive cost.

better and better estimates for values of N which are at least 10 times smaller than $N = 10^5$.