Statistics and Computing (2023) 33:82
https://doi.org/10.1007/s11222-023-10250-2

ORIGINAL PAPER

Automatically adapting the number of state particles in SMC^2

Imke Botha[1,3,4] · Robert Kohn[2,3,5] · Leah South[1,3,4] · Christopher Drovandi[1,3,4]

Received: 27 January 2022 / Accepted: 24 April 2023 / Published online: 29 May 2023

Abstract

Sequential Monte Carlo squared (SMC^2) methods can be used for parameter inference of intractable likelihood state-space models. These methods replace the likelihood with an unbiased particle filter estimate, similarly to particle Markov chain Monte Carlo (MCMC). As with particle MCMC, the efficiency of SMC^2 greatly depends on the variance of the likelihood estimator, and therefore on the number of state particles used within the particle filter. We introduce novel methods to adaptively select the number of state particles within SMC^2 using the expected squared jumping distance to trigger the adaptation, and modifying the exchange importance sampling method of Chopin et al. (J R Stat Soc: Ser B (Stat Method) 75(3):397–426, 2012) to replace the current set of state particles with the new set of state particles. The resulting algorithm is fully automatic, and can significantly improve current methods. Code for our methods is available at https://github.com/imkebotha/adaptive-exact-approximate-smc.

Keywords Bayesian inference · State-space models · SMC · Pseudo-marginal · Particle MCMC

1 Introduction

We are interested in exact Bayesian parameter inference for state-space models (SSMs) where the likelihood function of the model parameters is intractable. SSMs are ubiquitous in engineering, econometrics and the natural sciences; see Cappé et al. (2005) and references therein for an overview. They are used when the process of interest is observed indirectly over time or space, i.e. they consist of a hidden or latent process $\{X_t\}_{t\geq 1}$ and an observed process $\{Y_t\}_{t\geq 1}$.

Particle Markov chain Monte Carlo (MCMC; Andrieu et al. 2010; Andrieu and Roberts 2009) methods such as particle marginal Metropolis-Hastings (PMMH) or particle Gibbs can be used for exact parameter inference of intractable likelihood SSMs. PMMH uses a particle filter estimator of the likelihood within an otherwise standard Metropolis-Hastings algorithm. Similarly, particle Gibbs uses a conditional particle filter to draw the latent states from their full conditional distribution, then updates the model parameters conditional on the latent states. Both PMMH and particle Gibbs are simulation consistent under mild conditions (Andrieu et al. 2010).

Chopin et al. (2012) and Duan and Fulop (2014) apply a similar approach to sequential Monte Carlo (SMC) samplers. SMC methods for static models (Chopin 2002; Del Moral et al. 2006) recursively sample through a sequence of distributions using a combination of reweighting, resampling and mutation steps. In the Bayesian setting, this sequence often starts at the prior and ends at the posterior distribution. For intractable likelihood SSMs, Chopin et al. (2012) and Duan and Fulop (2014) replace the likelihood within the sequence of distributions being traversed with its unbiased estimator. Practically, this means that each parameter particle is augmented with N_x state particles. Due to this nesting of SMC algorithms and following Chopin et al. (2012), we refer to these methods as SMC^2. As with particle MCMC, for any fixed number of state particles (N_x), SMC^2 targets the exact posterior distribution (Duan and Fulop 2014).

We define an 'exact' method as one that converges to the true posterior distribution as the number of parameter samples (N_θ) goes to infinity (with finite N_x), with no extra assumptions above those required for standard MCMC

✉ Imke Botha
imke.botha@hdr.qut.edu.au

[1] School of Mathematical Sciences, Queensland University of Technology (QUT), Brisbane, Australia

[2] School of Economics, University of New South Wales, Sydney, Australia

[3] Australian Research Council Centre of Excellence for Mathematical & Statistical Frontiers, Melbourne, Australia

[4] QUT Centre for Data Science, Brisbane, Australia

[5] DARE: ARC Training Centre in Data Analytics for Resources and Environments, Sydney, Australia

or SMC. While similar methods to SMC^2 are available for Bayesian parameter inference of intractable likelihood SSMs, e.g. nested particle filters (Crisan and Míguez 2017, 2018) and ensemble MCMC (Drovandi et al. 2022), they are not exact in general and so are not considered in this paper. In particular, nested particle filters target the exact posterior as $N_\theta \to \infty$ and $N_x \to \infty$ subject to some assumptions about the optimal filter and the parameter space. Similarly, ensemble MCMC is guaranteed to be exact only when the model is linear Gaussian.

The sampling efficiency of particle MCMC and SMC^2 greatly depends on the number of state particles used within the particle filter. In particle MCMC, N_x is generally tuned manually, which can be time intensive. A significant advantage of SMC^2 over particle MCMC is that N_x can be adapted automatically. Strategies to do this are proposed by Chopin et al. (2012, 2015) and Duan and Fulop (2014); however, these methods automate the adaptation of N_x at the expense of other model-specific tuning parameters, which must then be tuned manually. Furthermore, the value of N_x can be difficult to choose in practice, and has a significant effect on both the Monte Carlo error of the SMC approximation to the target distribution and the computation time. Current methods require a moderate starting value of N_x to avoid poor values in subsequent iterations, i.e. values that are too low and negatively impact the accuracy of the samples, or unnecessarily high values that increase the computation time.

Adaptation of the number of state particles is also studied outside the SMC^2 context. Bhadra and Ionides (2016) propose an optimal allocation method, which uses a meta-model to estimate the variance of the incremental log-likelihood estimators. This method allocates the number of particles to be used for each timepoint in the particle filter. However, it still requires the total number of state particles (over all timepoints) to be known. Lee and Whiteley (2018) run successive particle filters, doubling the number of state particles each time, until the variance of the log-likelihood estimator is below some threshold. In the SMC^2 context, this approach can be very expensive computationally, as it needs to be applied to each parameter particle. Other methods adapt the number of state particles within the particle filter itself (Fox 2003; Soto 2005; Elvira et al. 2017, 2021), leading to a random number of state particles whenever the particle filter is run. This is problematic in the context of particle MCMC (and hence in the mutation step of SMC^2) as the dimension of the augmented parameter space changes whenever the likelihood is estimated. A key point of particle MCMC is that it can be reformulated as standard MCMC on an augmented space (Andrieu and Roberts 2009; Andrieu et al. 2010). To sample from the posterior on a space of varying dimension requires methods such as reversible jump MCMC (Green 1995).

Our article introduces a novel and principled strategy to automatically tune N_x, while aiming to keep an optimal balance between statistical and computational efficiency. Compared to current methods, our approach has less tuning parameters that require manual calibration. Notably, it also allows N_x to decrease, which makes our approach more robust to variability in the algorithm and the adaptation step. We find that using the expected squared jumping distance of the mutation step to adapt the number of state particles generally gives the most efficient and reliable results. To further improve the overall efficiency of the adaptation, we also modify the exchange importance sampling method of Chopin et al. (2012) to update the set of state particles once N_x is adapted. This modified version introduces no extra variability in the parameter particle weights, and outperforms the current methods.

The rest of the paper is organized as follows. Section 2 gives the necessary background on state-space models and SMC methods, including particle filters, SMC for static models and SMC^2. Section 3 describes the current methods for adapting the number of state particles in SMC^2. Section 4 describes our novel tuning methodology. Section 5 shows the performance of our methods on a Brownian motion model, a stochastic volatility model, a noisy theta-logistic model and a noisy Ricker model. Section 6 concludes.

2 Background

This section contains the necessary background information for understanding the novel methods discussed in Sect. 4. It covers content related to exact Bayesian inference for state-space models, particularly focussed on models with intractable transition densities.

2.1 State-space models

Consider a state-space model (SSM) with parameters $\boldsymbol{\theta} \in \Theta$, a hidden or latent process $\{X_t\}_{t\geq 1}$ and an observed process $\{Y_t\}_{t\geq 1}$. A key assumption of SSMs is that the process $\{(X_t, Y_t), t \geq 1\}$ is Markov, and we further assume that the full conditional densities of $Y_t = y_t$ and $X_t = x_t$ are

$$p(y_t \mid x_t, x_{t-1}, y_{t-1}, \boldsymbol{\theta}) = g(y_t \mid x_t, \boldsymbol{\theta}),$$

and

$$p(x_t \mid x_{t-1}, y_{t-1}, \boldsymbol{\theta}) = f(x_t \mid x_{t-1}, \boldsymbol{\theta}),$$

where $g(y_t \mid x_t, \boldsymbol{\theta})$ and $f(x_t \mid x_{t-1}, \boldsymbol{\theta})$ are the observation density and transition density respectively. The density of the latent states at time $t = 1$ is $\mu(x_1 \mid \boldsymbol{\theta})$ and the prior density of the parameters is $p(\boldsymbol{\theta})$.

Define $z_{i:j} := \{z_i, z_{i+1}, \ldots, z_j\}$ for $j \geq i$. The distribution of $\boldsymbol{\theta}$ conditional on the observations up to time $t \leq T$

is

$$p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:t}) = \frac{p(\boldsymbol{\theta})}{p(\boldsymbol{y}_{1:t})} \int_{\boldsymbol{x}_{1:t}} p(\boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t} \mid \boldsymbol{\theta}) d\boldsymbol{x}_{1:t}, \tag{1}$$

where

$$p(\boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t} \mid \boldsymbol{\theta}) = \mu(x_1 \mid \boldsymbol{\theta}) \prod_{i=2}^{t} f(x_i \mid x_{i-1}, \boldsymbol{\theta}) \prod_{i=1}^{t} g(y_i \mid x_i, \boldsymbol{\theta}). \tag{2}$$

The integral in (1) gives the likelihood function $p(\boldsymbol{y}_{1:t} \mid \boldsymbol{\theta})$. This integral is often analytically intractable or prohibitively expensive to compute, which means that the likelihood is also intractable. If the value of $\boldsymbol{\theta}$ is fixed, a particle filter targeting $p(\boldsymbol{x}_{1:t} \mid \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ gives an unbiased estimate of the likelihood as a by-product, as described in Sect. 2.2.1. Similarly, a conditional particle filter (Andrieu et al. 2010), i.e. a particle filter that is conditional on a single state trajectory $\boldsymbol{x}^k_{1:t}$, can be used to unbiasedly simulate latent state trajectories from $p(\cdot \mid \boldsymbol{x}^k_{1:t}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$. Particle filters are SMC methods applied to dynamic models.

2.2 Sequential Monte Carlo

SMC methods recursively sample from a sequence of distributions, $\pi_d(z_d) \propto \gamma_d(z_d)$, $d = 0, \ldots, D$, where $\pi_0(z_0)$ can generally be sampled from directly and $\pi_D(z_D)$ is the target distribution (Del Moral et al. 2006).

These distributions are traversed using a combination of resample, mutation and reweight steps. Initially, N_z samples are drawn from $\pi_0(z_0)$ and given equal weights $\{z^n_0, W^n_0 = {}^1/_{N_z}\}^{N_z}_{n=1}$. For each subsequent distribution, the particles are resampled according to their weights, thus removing particles with negligible weights and duplicating high-weight particles. The resampled particles are then mutated using R applications of the mutation kernel $K(z^n_{d-1}, z^n_d)$, and reweighted as

$$w^n_d = N_z^{-1} \cdot \frac{\gamma_d(z^n_d) L(z^n_d, z^n_{d-1})}{\gamma_{d-1}(z^n_{d-1}) K(z^n_{d-1}, z^n_d)}, \quad W^n_d = \frac{w^n_d}{\sum_{i=1}^{N_z} w^i_d},$$

where $L(z^n_d, z^n_{d-1})$ is the artificial backward kernel of Del Moral et al. (2006). Note that if the weights at iteration d are independent of the mutated particles z^n_d, the reweighting step should be completed prior to the resample and mutation steps. At each iteration d, the weighted particles $\{z^n_d, W^n_d\}^{N_z}_{n=1}$ form an approximation of $\pi_d(z_d)$. See Del Moral et al. (2006) for more details.

An advantage of SMC methods is that an unbiased estimate of the normalizing constant of the target distribution can be obtained as follows (Del Moral et al. 2006)

$$\int \gamma_D(z_D) dz \approx \prod_{d=0}^{D} \sum_{n=1}^{N_z} w^n_d. \tag{3}$$

This feature is exploited in the SMC^2 methods described in Sect. 2.3.

2.2.1 Particle filters

SMC methods for dynamic models are known as particle filters. For fixed $\boldsymbol{\theta}$, the sequence of filtering distributions for $d = 1, \ldots, T$, is

$$\begin{aligned}\pi_d(z_d) &:= p(\boldsymbol{x}_{1:d} \mid \boldsymbol{y}_{1:d}, \boldsymbol{\theta}) \\ &= \frac{\mu(x_1 \mid \boldsymbol{\theta})}{p(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta})} \prod_{i=2}^{d} f(x_i \mid x_{i-1}, \boldsymbol{\theta}) \prod_{i=1}^{d} g(y_i \mid x_i, \boldsymbol{\theta}).\end{aligned}$$

The bootstrap particle filter of Gordon et al. (1993) uses the transition density as the mutation kernel $K(x_{d-1}, x_d) = f(x_d \mid x_{d-1}, \boldsymbol{\theta})$, and selects $L(x_d, x_{d-1}) = 1$ as the backward kernel. The weights are then given by

$$w^m_d = N_x^{-1} g(y_d \mid x_d, \boldsymbol{\theta}), \quad W^m_d = \frac{w^m_d}{\sum_{i=1}^{N_x} w^i_d},$$

for $m = 1, \ldots, N_x$. Algorithm 1 shows pseudo-code for the bootstrap particle filter (Gordon et al. 1993).

Define $x^{1:N_x}_{1:d} := \{x^{1:N_x}_1, \ldots, x^{1:N_x}_d\}$, where $d = 1, \ldots, T$. The likelihood estimate with N_x state particles and d observations is then

$$\begin{aligned}\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}, \boldsymbol{x}^{1:N_x}_{1:d}) &= \prod_{i=1}^{d} \sum_{m=1}^{N_x} w^m_i \\ &= \prod_{i=1}^{d} \left(\frac{1}{N_x} \sum_{m=1}^{N_x} g(y_i \mid x^m_i, \boldsymbol{\theta}) \right).\end{aligned} \tag{4}$$

Let $\psi(\boldsymbol{x}^{1:N_x}_{1:d})$ be the joint distribution of all the random variables drawn during the course of the particle filter (Andrieu et al. 2010). The likelihood estimate in (4) is unbiased in the sense that $\mathbb{E}_{\psi(\boldsymbol{x}^{1:N_x}_{1:d})} \left(\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}, \boldsymbol{x}^{1:N_x}_{1:d}) \right) = p(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta})$ (Section 7.4.2 of Del Moral, 2004; see also Pitt et al. 2012).

The notation

$$\begin{aligned}\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}) &= \widehat{p_{N_x}}(\boldsymbol{y}_{1:d}, \boldsymbol{x}^{1:N_x}_{1:d} \mid \boldsymbol{\theta}) \\ &= \widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}, \boldsymbol{x}^{1:N_x}_{1:d}) \psi(\boldsymbol{x}^{1:N_x}_{1:d}) \\ &= \frac{1}{N_x} \sum_{m=1}^{N_x} \widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}, \boldsymbol{x}^m_{1:d}), \quad \boldsymbol{x}^m_{1:d} \sim \psi(\boldsymbol{x}^m_{1:d}),\end{aligned}$$

is used interchangeably throughout the paper.

Algorithm 1 The bootstrap particle filter of Gordon et al. (1993). The index (m) means 'for all $m \in \{1, \ldots, N_x\}$'

```
Input: data $\boldsymbol{y}_{1:d}$, number of state particles $N_x$ and the static parameters $\boldsymbol{\theta}$.
Output: likelihood estimate $\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta})$, set of weighted state particles $\{\boldsymbol{x}_{1:d}^{1:N_x}, \boldsymbol{W}_{1:d}^{1:N_x}\}$
/* Initialise (t=1) */
Initialise $x_1^{1:N_x} \sim \mu(\cdot \mid \boldsymbol{\theta})$ and calculate the initial weights
    $w_1^{(m)} = N_x^{-1} \cdot g(y_1 \mid x_1^{(m)}, \boldsymbol{\theta}), \quad W_1^{(m)} = \frac{w_1^{(m)}}{\sum_{i=1}^{N_x} w_1^i}$
/* Initialise likelihood estimate */
Initialise the likelihood estimate $\widehat{p_{N_x}}(\boldsymbol{y}_1 \mid \boldsymbol{\theta}) = \sum_{m=1}^{N_x} w_1^m$
for $t = 2$ to $d$ do
    /* Resample */
    Resample $N_x$ particles from $\boldsymbol{x}_{t-1}^{1:N_x}$ with probability $\boldsymbol{W}_{t-1}^{1:N_x}$
    /* Simulate forward */
    Simulate the particles forward, $x_t^{(m)} \sim f(\cdot \mid x_{t-1}^{(m)}, \boldsymbol{\theta})$
    /* Reweight */
    Re-weight the particles from $\pi_{t-1}(\cdot)$ to $\pi_t(\cdot)$
        $w_t^{(m)} = \frac{1}{N_x} \cdot g(y_t \mid x_t^{(m)}, \boldsymbol{\theta}), \quad W_t^{(m)} = \frac{w_t^{(m)}}{\sum_{i=1}^{N_x} w_t^i}$
    /* Update likelihood estimate */
    Update the likelihood estimate $\widehat{p_{N_x}}(\boldsymbol{y}_{1:t} \mid \boldsymbol{\theta}) = \widehat{p_{N_x}}(\boldsymbol{y}_{1:t-1} \mid \boldsymbol{\theta}) \cdot \sum_{m=1}^{N_x} w_t^m$
end for
```

2.2.2 SMC for static models

For static models, where inference on $\boldsymbol{\theta}$ is of interest, the sequence of distributions traversed by the SMC algorithm is $\pi_d(\boldsymbol{\theta}_d) \propto \gamma_d(\boldsymbol{\theta}_d)$, $d = 0, \ldots, D$, where $\pi_0(\boldsymbol{\theta}_0) = p(\boldsymbol{\theta})$ is the prior and $\pi_D(\boldsymbol{\theta}_D) = p(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:T})$ is the posterior distribution. Assuming that the likelihood function is tractable, there are at least two general ways to construct this sequence,

1. likelihood tempering, which gives $\pi_d(\boldsymbol{\theta}) \propto p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})^{g_d} p(\boldsymbol{\theta})$ for $d = 0, \ldots, D$, and where $0 = g_0 \leq \cdots \leq g_D = 1$, and
2. data annealing (Chopin 2002), which gives $\pi_d(\boldsymbol{\theta}) \propto p(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$ for $d = 0, \ldots, T$, where T is the number of observations and $D = T$.

Typically, SMC for static models uses a mutation kernel which ensures that the current target $\pi_d(\boldsymbol{\theta})$ remains invariant. A common choice is to use R applications of an MCMC mutation kernel along with the backward kernel $L(\boldsymbol{\theta}_d, \boldsymbol{\theta}_{d-1}) = \gamma_d(\boldsymbol{\theta}_{d-1}) K(\boldsymbol{\theta}_{d-1}, \boldsymbol{\theta}_d) / \gamma_d(\boldsymbol{\theta}_d)$ (Chopin 2002; Del Moral et al. 2006). The weights then become

$$w_d^n = N_\theta^{-1} \cdot \frac{\gamma_d(\boldsymbol{\theta}_{d-1}^n)}{\gamma_{d-1}(\boldsymbol{\theta}_{d-1}^n)}, \quad W_d^n = \frac{w_d^n}{\sum_{i=1}^{N_\theta} w_d^i}. \tag{5}$$

Since the weights are independent of the mutated particles $\boldsymbol{\theta}_d$, the reweighting step is completed prior to the resample and mutation steps.

2.3 SMC^2

Standard SMC methods for static models cannot be applied directly to state-space models if the parameters $\boldsymbol{\theta}$ are unknown except when the integral in (1) is analytically tractable. When the likelihood is intractable, SMC^2 replaces it in the sequence of distributions being traversed with a particle filter estimator. Essentially, each parameter particle is augmented with a set of weighted state particles.

Since the likelihood is replaced with a particle filter estimator, the parameter particles in SMC^2 are mutated using R applications of a particle MCMC mutation kernel $K(\cdot, \cdot)$. Section 2.4 describes the particle marginal Metropolis-Hastings (PMMH) algorithm. As with SMC for static models, the parameter particle weights are given by (5).

Two general ways to construct the sequence of targets for SMC^2 are the density tempered marginalised SMC algorithm of Duan and Fulop (2014) and the data annealing SMC^2 method of Chopin et al. (2012), which we refer to as density tempering SMC^2 (DT-SMC^2) and data annealing SMC^2 (DA-SMC^2) respectively. These are described in Sects. 2.3.1 and 2.3.2.

Algorithm 2 shows pseudo-code which applies to both DT-SMC^2 and DA-SMC^2. The main difference between the two methods is how the sequence of targets is defined. Sections 2.3.1 and 2.3.2 describe the sequence of targets and the reweighting formulas for DT-SMC^2 and DA-SMC^2 respectively. For conciseness, we denote the set of weighted state particles associated with parameter particle $n, n = 1, \ldots, N_\theta$ at iteration d as

$$\tilde{\boldsymbol{x}}_d^{1:N_x,n} := \begin{cases} \{\boldsymbol{x}_{1:T}^{1:N_x,n}, \boldsymbol{S}_d^{1:N_x,n}\}, & \text{for DT-SMC}^2, \\ \{\boldsymbol{x}_{1:d}^{1:N_x,n}, \boldsymbol{S}_d^{1:N_x,n}\}, & \text{for DA-SMC}^2, \end{cases}$$

where $\boldsymbol{S}_d^{1:N_x,n}$ is the set of normalised state particle weights. The nth parameter particle with its attached set of weighted state particles is denoted as $\boldsymbol{\vartheta}_d^n = \{\boldsymbol{\theta}_d^n, \tilde{\boldsymbol{x}}_d^{1:N_x,n}\}$, $n = 1, \ldots, N_\theta$.

2.3.1 Density tempering SMC^2

The sequence of distributions for DT-SMC^2 is

$$\pi_d(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}) \left[\widehat{p_{N_x}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}, \boldsymbol{x}_{1:T}^{1:N_x})\right]^{g_d} \psi(\boldsymbol{x}_{1:T}^{1:N_x}),$$
$$0 = g_0 \leq \cdots \leq g_D = 1,$$

which gives the weights from (5) as

$$w_d^n = N_\theta^{-1} \cdot \left[\widehat{p_{N_x}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_{d-1}^n, \boldsymbol{x}_{1:T}^{1:N_x})\right]^{g_d - g_{d-1}},$$

$$W_d^n = \frac{w_d^n}{\sum_{i=1}^{N_\theta} w_d^i}. \tag{6}$$

Due to the tempering parameter g_d, DT-SMC^2 is only exact at the first and final temperatures, i.e. $p(\boldsymbol{\theta})p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})^{g_d} / \int p(\boldsymbol{\theta})p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta})^{g_d} d\boldsymbol{\theta}$ is a marginal distribution of $\pi_d(\boldsymbol{\theta})$ only at $g_1 = 0$ and $g_D = 1$.

2.3.2 Data annealing SMC^2

For DA-SMC^2, the sequence of distributions is

$$\pi_d(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta})\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}, \boldsymbol{x}_{1:d}^{1:N_x})\psi(\boldsymbol{x}_{1:d}^{1:N_x}), \quad D = T,$$

and the weights from (5) are

$$w_d^n = N_\theta^{-1} \cdot \widehat{p_{N_x}}\left(y_d \mid \boldsymbol{y}_{1:d-1}, \boldsymbol{\theta}_{d-1}^n\right), \quad W_d^n = \frac{w_d^n}{\sum_{i=1}^{N_\theta} w_d^i}, \tag{7}$$

where $\widehat{p_{N_x}}\left(y_d \mid \boldsymbol{y}_{1:d-1}, \boldsymbol{\theta}_{d-1}^n\right)$ is obtained from iteration d of a particle filter (see (4) and Algorithm 1). Unlike DT-SMC^2, DA-SMC^2 admits $p\left(\boldsymbol{\theta} \mid \boldsymbol{y}_{1:d}\right)$ as a marginal distribution of $\pi_d(\boldsymbol{\theta})$ for all $d = 0, \dots, D$.

2.4 Particle MCMC mutations

The simplest mutation of the parameter particles in SMC^2 is a sequence of Markov move steps using the PMMH algorithm; see Gunawan et al. (2021) for alternatives. The PMMH method is a standard Metropolis-Hastings algorithm where the intractable likelihood is replaced by the particle filter estimate in (4). Algorithm 3 shows a single PMMH iteration.

While a PMMH mutation leaves the current target invariant, its acceptance rate is sensitive to the variance of the likelihood estimator (Andrieu et al. 2010). In practice, this means that if the variance is too high, then some particles may not be mutated during the mutation step—even with a large number of MCMC iterations.

In the context of particle MCMC samplers, Andrieu et al. (2010) show that N_x must be chosen as $\mathcal{O}(T)$ to achieve reasonable acceptance rates, i.e. reasonable variance of the likelihood estimator. Pitt et al. (2012), Doucet et al. (2015) and Sherlock et al. (2015) recommend choosing N_x such that the variance of the log-likelihood estimator is between 1 and 3 when evaluated at, e.g., the posterior mean. This generally requires a (potentially time-consuming) tuning process for N_x before running the algorithm.

Algorithm 2 The SMC^2 Algorithm. The index (n) means 'for all $n \in \{1, \dots, N_\theta\}$'

```
Input: data $\boldsymbol{y}_{1:T}$, number of parameter particles $N_\theta$, number of state particles $N_x$, number of MCMC iterations $R$
Output: set of weighted particles $\{\boldsymbol{\vartheta}_D^{1:N_{\boldsymbol{\theta}}}, \boldsymbol{W}_D^{1:N_{\boldsymbol{\theta}}}\}$
/* Initialisation step (t=0) */
Initialise $\boldsymbol{\vartheta}_0^{1:N_\theta}$ and set $W_0^{(n)} = \frac{1}{N_\theta}$
for $d = 1$ to $D$ do
    /* Reweight */
    Re-weight the particles from $\pi_{d-1}(\cdot)$ to $\pi_d(\cdot)$ using (6) or (7).
    /* Resample */
    Resample $N_{\boldsymbol{\theta}}$ particles from $\boldsymbol{\vartheta}_d^{1:N_\theta}$ with probability $\boldsymbol{W}_d^{1:N_\theta}$
    /* Mutate */
    for $r = 1$ to $R$ do
        PMMH mutation $\boldsymbol{\vartheta}_d^{(n)} \sim K\left(\boldsymbol{\vartheta}_d^{(n)}, \cdot\right)$ (See Algorithm 3)
    end for
end for
```

Algorithm 3 A single iteration of the particle marginal Metropolis-Hastings algorithm.

```
Input: data $\boldsymbol{y}$, proposal distribution $q(\cdot)$, current parameter value $\boldsymbol{\theta}_d$, current likelihood estimate $\widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}_d)$. Note that $\boldsymbol{y} := \boldsymbol{y}_{1:T}$ for DT-SMC$^2$and $\boldsymbol{y} := \boldsymbol{y}_{1:d}$ for DA-SMC$^2$. Optional: current set of weighted state particles $\tilde{\boldsymbol{x}}_d^{1:N_x}$
Output: new parameter value $\boldsymbol{\theta}_d$, new likelihood estimate $\widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}_d)$. Optional: new set of weighted state particles $\tilde{\boldsymbol{x}}_d^{1:N_x}$
Sample $\boldsymbol{\theta}_d^* \sim q(\cdot \mid \boldsymbol{\theta}_d)$,
Run Algorithm 1 to obtain $\widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}_d^*)$ and $\tilde{\boldsymbol{x}}_d^{1:N_x,*}$,
Calculate acceptance probability
    $\alpha(\boldsymbol{\theta}_d, \boldsymbol{\theta}_d^*) = \min\left(1, \frac{\widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}_d^*)p(\boldsymbol{\theta}_d^*)}{\widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}_d)p(\boldsymbol{\theta}_d)} \frac{q(\boldsymbol{\theta}_d \mid \boldsymbol{\theta}_d^*)}{q(\boldsymbol{\theta}_d^* \mid \boldsymbol{\theta}_d)}\right).$    (8)
With probability $\alpha(\boldsymbol{\theta}_d, \boldsymbol{\theta}_d^*)$, set
    $\boldsymbol{\theta}_d = \boldsymbol{\theta}_d^*, \quad \widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}_d) = \widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}_d^*), \quad \tilde{\boldsymbol{x}}_d^{1:N_x} = \tilde{\boldsymbol{x}}_d^{1:N_x,*},$
    otherwise keep the current values of $\boldsymbol{\theta}_d$, $\widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}_d)$ and $\tilde{\boldsymbol{x}}_d^{1:N_x}$.
```

For SMC^2, fewer particles may be required to achieve reasonable acceptance rates in the early stages of the algorithm. In DA-SMC^2, $N_x = \mathcal{O}(t)$, where $t = d$, suggests starting with a small N_x, and increasing it with each added observation. Likewise, in DT-SMC^2, a small g_d will reduce the impact of a highly variable log-likelihood estimator. In addition, unlike particle MCMC methods, it is possible to automatically adapt N_x within SMC^2. The next section describes the tuning strategies proposed by Chopin et al. (2012, 2015) and Duan and Fulop (2014).

3 Existing methods to calibrate N_x within SMC^2

There are three main stages to adapting N_x: (1) triggering the adaptation, (2) choosing the new number of particles N_x^*, and (3) replacing the current set of state particles $\tilde{\boldsymbol{x}}_d^{1:N_x,1:N_\theta}$ with the new set $\tilde{\boldsymbol{x}}_d^{1:N_x^*,1:N_\theta}$. To simplify notation, we write $\tilde{\boldsymbol{x}}_d^{1:N_x,1:N_\theta}$ as $\tilde{\boldsymbol{x}}_d^{1:N_x}$.

Stage 1: Triggering the adaptation

It may be necessary to adapt N_x when the mutation step no longer achieves sufficient particle diversity. Chopin et al. (2012, 2015) and Duan and Fulop (2014) fix the number of MCMC iterations (R) and change N_x whenever the acceptance rate of a single MCMC iteration falls below some target value. This approach has two main drawbacks. First, the acceptance rate does not take the jumping distances of the particles into account, and can be made artificially high by making very local proposals. Second, both R and the target acceptance rate must be tuned—even if the exact likelihood is used, the acceptance rate may naturally be low, depending on the form of the posterior and the proposal function used within the mutation kernel. Ideally, N_x and R should be jointly adapted.

Stage 2: Choosing the new number of particles N_x^*

A new number of state particles (N_x^*) is determined in the second stage. Chopin et al. (2012) set $N_x^* = 2 \cdot N_x$ (DOUBLE), while Duan and Fulop (2014) set $N_x^* = \widehat{\sigma_{N_x}}^2 \cdot N_x$ (RESCALE- VAR), where $\widehat{\sigma_{N_x}}^2$ is the estimated variance of the log-likelihood estimator using N_x state particles. The variance is estimated from k independent estimates of the log-likelihood (for the current SMC target) based on the sample mean of the parameter particles. This choice is motivated by the results of Pitt et al. (2012), Doucet et al. (2015) and Sherlock et al. (2015), who show that $\sigma^2_{N_x} \propto 1/N_x$ for any number of state particles N_x. Setting $\sigma^2_{N_x} = \alpha/N_x$ and rearranging gives both $\alpha = \sigma^2_{N_x} \cdot N_x$ and $N_x = \alpha/\sigma^2_{N_x}$. Given N_x and $\sigma^2_{N_x}$, these expressions can be used to find a new number of state particles N_x^* such that $\sigma^2_{N_x^*} = 1$, by noting that $N_x^* = \alpha/\sigma^2_{N_x^*} = \alpha/1 = \sigma^2_{N_x} \cdot N_x$.

We find that if the initial N_x is too small, then the DOUBLE scheme of Chopin et al. (2012) can take a significant number of iterations to set N_x to a reasonable value. It can also increase N_x to an unnecessarily high value if the adaptation is triggered when the number of state particles is already large.

While the RESCALE- VAR method of Duan and Fulop (2014) is more principled, as it takes the variance of the log-likelihood estimator into account, we find that it is also sensitive to the initial number of particles. For a poorly chosen initial N_x, the variance of the log-likelihood estimator can be of order 10^2 or higher. In this case, scaling the current number of particles by $\widehat{\sigma_{N_x}}^2$ may give an extremely high value for N_x^*.

Chopin et al. (2015) propose a third method; they set $N_x^* = \tau/\sigma^2_{N_x}$, where τ is a model-specific tuning parameter, and $\sigma^2_{N_x}$ is the variance of the log-likelihood estimator with N_x state particles. This choice is motivated by the results from Doucet et al. (2012) (an earlier version of Doucet et al. (2015)); see Chopin et al. (2015) for further details. Since the parameter τ must be tuned manually, this approach is not included in our numerical experiments in Sect. 5.

Stage 3: Replacing the state particle set

The final stage replaces the current set of state particles $\tilde{\boldsymbol{x}}_d^{1:N_x}$ by the new set $\tilde{\boldsymbol{x}}_d^{1:N_x^*}$. Chopin et al. (2012) propose a reweighting step for the parameter particles (REWEIGHT) using the generalised importance sampling method of Del Moral et al. (2006) to swap $\tilde{\boldsymbol{x}}_d^{1:N_x}$ with $\tilde{\boldsymbol{x}}_d^{1:N_x^*}$. The incremental weight function for this step (for DA-SMC^2) is

$$
\begin{aligned}
IW &= \frac{\pi_d\left(\boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x^*} \mid \boldsymbol{y}_{1:d}\right) L_d(\boldsymbol{x}_d^{1:N_x^*}, \boldsymbol{x}_d^{1:N_x})}{\pi_d\left(\boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x} \mid \boldsymbol{y}_{1:d}\right) \psi(\boldsymbol{x}_d^{1:N_x^*})} \\
&= \frac{p(\boldsymbol{\theta}_d)\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x^*})\psi(\boldsymbol{x}_d^{1:N_x^*})L_d(\boldsymbol{x}_d^{1:N_x^*}, \boldsymbol{x}_d^{1:N_x})}{p(\boldsymbol{\theta}_d)\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x})\psi(\boldsymbol{x}_d^{1:N_x})\psi(\boldsymbol{x}_d^{1:N_x^*})} \\
&= \frac{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x^*})L_d(\boldsymbol{x}_d^{1:N_x^*}, \boldsymbol{x}_d^{1:N_x})}{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x})\psi(\boldsymbol{x}_d^{1:N_x})},
\end{aligned}
$$

where $L_d(\boldsymbol{x}_d^{1:N_x^*}, \boldsymbol{x}_d^{1:N_x})$ is the backward kernel. They use the following approximation to the optimal backward kernel (see Proposition 1 of Del Moral et al. (2006))

$$
\begin{aligned}
L_d(\boldsymbol{x}_d^{1:N_x^*}, \boldsymbol{x}_d^{1:N_x}) &= \frac{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x})\psi(\boldsymbol{x}_d^{1:N_x})}{p(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d)} \\
&\approx \frac{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x})\psi(\boldsymbol{x}_d^{1:N_x})}{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x})} = \psi(\boldsymbol{x}_d^{1:N_x}),
\end{aligned}
\qquad (9)
$$

leading to

$$
IW_d = \frac{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x^*})}{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x})}.
$$

For density tempering, this becomes

$$IW_d = \left(\frac{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x^*})}{\widehat{p_{N_x}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_d^{1:N_x})} \right)^{g_d}.$$

The new parameter particle weights are then given by

$$w_d^n = W_{d-1}^n \cdot IW_d^n, \quad W_d^n = \frac{w_d^n}{\sum_{i=1}^{N_\theta} w_d^i}.$$

While this method is relatively fast, it can significantly increase the variance of the parameter particle weights (Duan and Fulop 2014).

As an alternative to REWEIGHT, Chopin et al. (2012) propose a conditional particle filter (CPF) step to replace $\tilde{\boldsymbol{x}}_d^{1:N_x}$ with $\tilde{\boldsymbol{x}}_d^{1:N_x^*}$. Here, the state particles and the likelihood estimates are updated by running a particle filter conditional on a single trajectory from the current set of state particles. The incremental weight function of this step is 1, which means that the parameter particle weights are left unchanged. The drawback of this approach is that all the state particles must be stored, which can significantly increase the RAM required by the algorithm. Chopin et al. (2015) propose two extensions of the CPF approach which reduce the memory requirements of the algorithm at the expense of increased computation time. Their first proposal is to only store the state particles with descendants at the final time-point, i.e. using a path storage algorithm within the particle filter (Jacob et al. 2015). Their second method is to store the random seed of the pseudo-random number generator in such a way that the latent states and their associated ancestral indices can be re-generated at any point. Both variants still have a higher RAM requirement and run time compared to the REWEIGHT method.

Duan and Fulop (2014) propose a reinitialisation scheme to extend the particles (REINIT). Whenever N_x is increased, they fit a mixture model $Q(\cdot)$ informed by the current set of particles, then reinitialise the SMC algorithm with N_x^* state particles and $Q(\cdot)$ as the initial distribution. The modified sequence of distributions for DT-SMC^2 is

$$\begin{aligned}&\pi_d(\boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x} \mid \boldsymbol{y}_{1:T})\\ &\quad\propto [Q(\boldsymbol{\theta}_d)]^{1-g_d}[p(\boldsymbol{\theta}_d)\widehat{p_{N_x}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x})]^{g_d}\psi(\boldsymbol{x}_{1:T}^{1:N_x}),\\ &\quad 0 = g_0 \le \cdots \le g_D = 1.\end{aligned}$$

The REINIT method aims to minimize the variance of the weights, but we find it can be very slow as the algorithm may reinitialise numerous times before completion, each time with a larger number of particles. This approach also assumes that the distribution of the set of parameter particles when REINIT is triggered is more informative than the prior, which is not necessarily the case if the adaptation is triggered early.

4 Methods

This section describes our proposed approach for each of the three stages involved in adapting the number of state particles.

4.1 Triggering the adaptation

Instead of using the acceptance rate to measure particle diversity, we use the expected squared jumping distance (ESJD), which accounts for both the acceptance rate (the probability that the particles will move) and the jumping distance (how far they will move). See Pasarica and Gelman (2010), Fearnhead and Taylor (2013), Salomone et al. (2018) and Bon et al. (2021) for examples of this idea outside the SMC^2 context. The ESJD at iteration d is defined as

$$\text{ESJD}_d = \mathbb{E}\left[\left\|\boldsymbol{\theta}_d^* - \boldsymbol{\theta}_d\right\|^2\right]$$

where $\left\|\boldsymbol{\theta}_d^* - \boldsymbol{\theta}_d\right\|^2$ is the squared Mahalanobis distance between the current value of the parameters ($\boldsymbol{\theta}_d$) and the proposed value ($\boldsymbol{\theta}_d^*$). The ESJD of the rth MCMC iteration of the mutation step at iteration d (steps 5–7 of Algorithm 2) can be estimated as

$$\widehat{\text{ESJD}}_{d,r} = \frac{1}{N_\theta}\sum_{n=1}^{N_\theta}(\boldsymbol{\theta}_{d,r}^n - \boldsymbol{\theta}_{d,r}^{n,*})^\top \widehat{\Sigma}^{-1}(\boldsymbol{\theta}_{d,r}^n - \boldsymbol{\theta}_{d,r}^{n,*})\alpha(\boldsymbol{\theta}_{d,r}^n, \boldsymbol{\theta}_{d,r}^{n,*}),$$

where $\boldsymbol{\theta}_{d,r}^n$ is the nth parameter particle at the start of the rth MCMC iteration, $\boldsymbol{\theta}_{d,r}^{n,*}$ is the proposed parameter particle at the rth MCMC iteration, $\widehat{\Sigma}$ is the covariance matrix of the current parameter particle set, and $\alpha(\boldsymbol{\theta}_{d,r}^n, \boldsymbol{\theta}_{d,r}^{n,*})$ is the acceptance probability in (8). The total estimated ESJD for iteration d is $\widehat{\text{ESJD}}_d = \sum_{r=1}^R \widehat{\text{ESJD}}_{d,r}$.

Algorithm 4 outlines how N_x and R are adapted. To summarise, the adaptation is triggered in iteration d if $\widehat{\text{ESJD}}_{d-1}$ is below some target value (stage 1). Once triggered, the number of particles is adapted (stage 2) and the particle set is updated (stage 3). A single MCMC iteration is then run with the new number of particles, and the results from this step are used to determine how many MCMC iterations are required to reach the target ESJD, i.e. R is given by dividing the target ESJD by the estimated ESJD of the single MCMC iteration and rounding up. Once the adaptation is complete, the remaining MCMC iterations are completed. This approach gives a general framework which can be implemented with any of the stage 2 and stage 3 methods described in Sect. 3, as well as our novel methods in Sects. 4.2 and 4.3.

Algorithm 4 Novel method to adapt the number of state particles and mutate the parameter particles for SMC^2.

```
Input: the estimated ESJD from the previous iteration ($\widehat{\mathrm{ESJD}}_{d-1}$), the target ESJD for each iteration ($\widehat{\mathrm{ESJD}}_{\mathrm{target}}$) and the current set of particles $\boldsymbol{\vartheta}_d^{1:N_\theta}$
Output: new number of state particles $N_x$, estimated ESJD ($\widehat{\mathrm{ESJD}}_d$) and mutated set of particles $\boldsymbol{\vartheta}_d^{1:N_\theta}$
  /* Trigger the adaptation */
adapt = $\widehat{\mathrm{ESJD}}_{d-1} < \widehat{\mathrm{ESJD}}_{\mathrm{target}}$
if adapt then
  /* Adapt $N_x$ */
    Set new $N_x$ and update the particle set using any combination of the stage 2 and stage 3 methods described in Sections 3, 4.2 and 4.3
end if
  /* Initial mutation step with updated $N_x$ (if applicable) */
PMMH mutation $\boldsymbol{\vartheta}_{d,1}^{1:N_\theta} \sim K(\boldsymbol{\vartheta}_d^{1:N_\theta}, \cdot)$, calculate $\widehat{\mathrm{ESJD}}_{d,1}$
if adapt then
    /* Adapt $R$ */
    Set $R = \left\lceil \widehat{\mathrm{ESJD}}_{\mathrm{target}} / \widehat{\mathrm{ESJD}}_{d,1} \right\rceil$
end if
  /* Remaining mutation steps */
for $r = 2$ to $R$ do
    PMMH mutation $\boldsymbol{\vartheta}_{d,r}^{1:N_\theta} \sim K(\boldsymbol{\vartheta}_{d,r-1}^{1:N_\theta}, \cdot)$
end for
Set $\boldsymbol{\vartheta}_d^{1:N_\theta} = \boldsymbol{\vartheta}_{d,R}^{1:N_\theta}$
```

4.2 Choosing the new number of particles N_x^*

To set the new number of state particles N_x^*, we build on the RESCALE- VAR method of Duan and Fulop (2014), which adapts the number of state particles as follows:

1. Calculate $\bar{\boldsymbol{\theta}}_d$, the mean of the current set of parameter samples $\boldsymbol{\theta}_d^{1:N_\theta}$.
2. Run the particle filter with N_x state particles k times to get k estimates of the log-likelihood evaluated at $\bar{\boldsymbol{\theta}}_d$.
3. Calculate $\widehat{\sigma_{N_x}}^2$, the sample variance of the k log-likelihood estimates.
4. Set the new number of state particles to $N_x^* = \widehat{\sigma_{N_x}}^2 \cdot N_x$.

Recall from Sect. 3, that RESCALE- VAR is based on the relation $\sigma_{N_x}^2 \propto 1/N_x$ (Pitt et al. 2012; Doucet et al. 2015; Sherlock et al. 2015). In practice, we find that it changes N_x too drastically from one iteration to the next for two reasons. First, the sample variance may itself be highly variable, especially when N_x is small. Second, the sample mean of the parameter particles changes throughout the iterations, meaning that the number of state particles needed to reach a variance of 1 also changes throughout the iterations. The sample mean may also be a poor value at which to estimate the likelihood if the current target is multimodal or if the current set of parameter particles offers a poor Monte Carlo approximation to the current target distribution. The latter may occur if the number of parameter particles N_θ is too low.

Our first attempt to overcome some of these problems is to scale the number of state particles by the standard deviation instead of the variance, i.e. we set $N_x^* = \widehat{\sigma_{N_x}} \cdot N_x$ and call this method RESCALE- STD. A variance of 1 is still the overall target, however, more moderate values of N_x are proposed when $\widehat{\sigma_{N_x}}^2 \neq 1$. At any given iteration, the new target variance is the current standard deviation, i.e. N_x^* is chosen such that $\widehat{\sigma_{N_x^*}}^2 = \widehat{\sigma_{N_x}}$. The main drawback of RESCALE- STD is that the variance at the final iteration may be too high, depending on the initial value of N_x and the variability of the sample variance between iterations, i.e. it may approach a variance of 1 too slowly. In our numerical experiments in Sect. 5, however, we find that the final variance of the RESCALE- STD method is generally between 1 and 1.2^2, which is fairly conservative. In their numerical experiments, Doucet et al. (2015) found that the optimal N_x generally gives a variance that is between $1.2^2 = 1.44$ and $1.5^2 = 2.25$.

Our second method (which we refer to as NOVEL- VAR) aims to improve upon RESCALE- VAR by estimating the variance at different values of N_x. To obtain our set of candidate values, $\boldsymbol{N}_{x,1:M}$, we scale N_x by different fractional powers of $\widehat{\sigma_{N_x}}^2/\sigma_{\mathrm{target}}^2$, where $\sigma_{\mathrm{target}}^2$ is the target variance. Note that the candidate values $\boldsymbol{N}_{x,1:M}$ will be close to N_x if $\widehat{\sigma_{N_x}}^2$ is close to $\sigma_{\mathrm{target}}^2$. To avoid unnecessary computation, the current N_x is left unchanged if $\widehat{\sigma_{N_x}}^2$ falls within some range $\sigma_{\mathrm{min}}^2 < \sigma_{\mathrm{target}}^2 < \sigma_{\mathrm{max}}^2$. We also round the candidate number of state particles up to the nearest 10, which ensures that there is at least a difference of 10 between each $N_{x,m} \in \boldsymbol{N}_{x,1:M}$. Once $\boldsymbol{N}_{x,1:M}$ has been obtained, the variance is estimated for each $N_{x,m} \in \boldsymbol{N}_{x,1:M}$, and the new number of state particles is set to the $N_{x,m}$ that has the highest variance less than or equal to σ_{max}^2. In our numerical experiments in Sect. 5, we set

$$\boldsymbol{N}_{x,1:3} = \left\lceil N_x \cdot \left\{ s^{0.5},\ s^{0.75},\ s \right\}^\top \right\rceil, \quad s = \frac{\widehat{\sigma_{N_x}}^2}{\sigma_{\mathrm{target}}^2},$$

which gives candidate values ranging from RESCALE- STD ($s^{0.5} \cdot N_x$) to RESCALE- VAR ($s^1 \cdot N_x$). The target, minimum and maximum variances are $\sigma_{\mathrm{target}}^2 = G \cdot 1$, $\sigma_{\mathrm{min}}^2 = G \cdot 0.95^2$ and $\sigma_{\mathrm{max}}^2 = G \cdot 1.05^2$ respectively, where $G = 1$ for DA-SMC^2 and $G = 1/\max(0.6^2, g_d^2)$ for DT-SMC^2. These values are fairly conservative and aim to keep the final variance between $0.95^2 \approx 0.9$ and $1.05^2 \approx 1.1$.

The parameter G is used to take advantage of the effect of the tempering parameter on the variance, i.e. $\mathrm{var}(\log(\widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta})^{g_d})) = g^2 \cdot \mathrm{var}(\log(\widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta})))$. Capping the value of G is necessary in practice, since aiming for an excessive variance is difficult due to the variability of the variance estimate when N_x is low. By setting $G = 1/\max(0.6^2, g_d^2)$, the highest variance targeted is $1/0.36 \approx 2.8$. In general, we recommend not aiming for a variance that is greater than 3 (Sherlock et al. 2015). Note that including the tempering parameter in this way is infeasible for RESCALE- VAR or RESCALE- STD. For the former, changing the target variance only exacerbates the problem of too drastic changes of N_x between iterations. This is largely due to the increased variability of the sample variance when $g_d < 1$. While the variability of $\widehat{\sigma_{N_x}}^2$ is less of a problem for RESCALE- STD, this method struggles keeping up with the changing variance target.

Compared to RESCALE- VAR, we find that both RESCALE- STD and NOVEL- VAR are significantly less sensitive to the initial number of state particles, sudden changes in the variance arising from changes in the sample mean of the parameter particles, and variability in the estimated variance of the log-likelihood estimator. The NOVEL- VAR method is also more predictable in what variance is targeted at each iteration compared to RESCALE- STD.

Our final method (NOVEL- ESJD) also compares different values of N_x, but using the ESJD instead of the variance of the log-likelihood estimator. As before, the choice of candidate values $\boldsymbol{N}_{x,1:M}$ is flexible; in the numerical experiments in Sect. 5, we set

$$\boldsymbol{N}_{x,1:4} = \left\lceil N_x \cdot \left\{1, 2, s^{0.5}, s^1\right\}^\top \right\rceil, \quad s = \frac{\widehat{\sigma_{N_x}}^2}{G}, \tag{10}$$

where $G = 1$ for DA-SMC^2 and $G = 1/\max(0.6^2, g_d^2)$ for DT-SMC^2. Again, each $N_{x,m} \in \boldsymbol{N}_{x,1:M}$ is rounded up to the nearest 10. A score is calculated for a particular $N_{x,m} \in \boldsymbol{N}_{x,1:M}$ by first doing a mutation step with $N_{x,m}$ state particles, then calculating the number of MCMC iterations (R_m) needed to reach the ESJD target; the score for $N_{x,m}$ is $(N_m \cdot R_m)^{-1}$. Algorithm 5 describes the adaptive mutation step when using NOVEL- ESJD. Since the candidate N_x values are tested in ascending order (see step 2 of Algorithm 5), it is unnecessary to continue testing the values once the score starts to decrease (steps 8–17 of Algorithm 5).

This method does not target a particular variance, but instead aims to select the N_x having the cheapest mutation while still achieving the ESJD target. Compared to DOUBLE and the variance-based methods, we find that NOVEL- ESJD is consistent between independent runs, in terms of the run time and the adaptation for N_x. It is also relatively insensitive to the initial number of state particles, as well as variability in the variance of the likelihood estimator.

Ideally, the adaptation algorithm (Algorithm 4 or Algorithm 5) will only be triggered if N_x or R is too low (or too high, as mentioned in Sect. 5). In practice, the ESJD is variable, so the adaptation may be triggered more often than necessary. Allowing the number of state particles to decrease helps to keep the value of N_x reasonable. Also, if the estimated variance is close to the target variance, one of the candidate N_x values will be close in value to the current N_x. See Table 1 for an example of the possible values of N_x for the different methods.

Algorithm 5 Novel method to adapt the number of state particles and mutate the parameter particles for SMC^2 when using NOVEL- ESJD.

Input: the estimated ESJD from the previous iteration ($\widehat{\text{ESJD}}_{d-1}$), the target ESJD for each iteration ($\widehat{\text{ESJD}}_{\text{target}}$) and the current set of particles $\boldsymbol{\vartheta}_d^{1:N_\theta}$

Output: new number of state particles N_x, estimated ESJD ($\widehat{\text{ESJD}}_d$) and mutated set of particles $\boldsymbol{\vartheta}_d^{1:N_\theta}$

```
1: if $\widehat{\text{ESJD}}_{d-1} < \widehat{\text{ESJD}}_{\text{target}}$ then
       /* Adapt $N_x$ and $R$ */
2:     Calculate the set of candidate values, $\boldsymbol{N}_{x,1:M}$ (e.q. using (10)),
       and sort in ascending order, such that $N_{x,1} < N_{x,2} < \ldots <$
       $N_{x,M}$. Set $m^* = M$.
3:     for $N_{x,m} \in \boldsymbol{N}_{x,1:M}$ do
4:         Replace the current set of state particles with $\tilde{\boldsymbol{x}}_d^{1:N_{x,m}}$ using
           the method described in Section 4.3
5:         PMMH mutation $\boldsymbol{\vartheta}_{d,m}^{1:N_\theta} \sim K(\boldsymbol{\vartheta}_d^{1:N_\theta}, \cdot)$, calculate $\widehat{\text{ESJD}}_{d,m}$
6:         Calculate $R_m = \left\lceil \widehat{\text{ESJD}}_{\text{target}} / \widehat{\text{ESJD}}_{d,m} \right\rceil$
7:         Calculate score $z_m = (N_{x,m} \cdot R_m)^{-1}$
           /* If more than one value has been tested */
8:         if $m > 1$ then
               /* If the current score is worse than the previous one */
9:             if $z_m / z_{m-1} < 1$ then
10:                Set $m^* = m - 1$
11:                Replace the current set of state particles with $\tilde{\boldsymbol{x}}_d^{1:N_{x,m^*}}$
                   using the method described in Section 4.3
12:                Break
               /* If the current score is equal to the previous one */
13:            else if $z_m / z_{m-1} = 1$ then
14:                Set $m^* = m$
15:                Break
16:            end if
17:        end if
18:    end for
       /* Update $N_x$ and $R$ */
19:    Set $N_x = N_{x,m^*}$ and $R = R_{m^*}$
20: else
       /* Initial mutation step */
21:    PMMH mutation $\boldsymbol{\vartheta}_{d,1}^{1:N_\theta} \sim K(\boldsymbol{\vartheta}_d^{1:N_\theta}, \cdot)$, calculate $\widehat{\text{ESJD}}_{d,1}$
22: end if
    /* Remaining mutation steps */
23: for $r = 2$ to $R$ do
24:    PMMH mutation $\boldsymbol{\vartheta}_{d,r}^{1:N_\theta} \sim K(\boldsymbol{\vartheta}_{d,r-1}^{1:N_\theta}, \cdot)$
25: end for
26: Set $\boldsymbol{\vartheta}_d^{1:N_\theta} = \boldsymbol{\vartheta}_{d,R}^{1:N_\theta}$
```

Table 1 Possible values of the number of state particles N_x if N_x is currently 100 and $G = 1$, where G accounts for the tempering parameter in DT-SMC^2

$\widehat{\sigma_{N_x}}^2$	Candidate values $\boldsymbol{N_x}$				
	DOUBLE	RESCALE- VAR	RESCALE- STD	NOVEL- VAR	NOVEL- ESJD
0.5	200	50	71	50, 60, 71	50, 71, 100, 200
1	200	100	100	100	100, 200
1.5	200	150	123	123, 136, 150	100, 123, 150, 200
50	200	5000	708	708, 1881, 5000	100, 200, 708, 5000

Note that we allow the number of particles to decrease with RESCALE- VAR. The new N_x will be one of the possible values listed, e.g. if $\widehat{\sigma_{N_x}}^2 = 1$, NOVEL- ESJD will set N_x to 100 or 200 depending on which value is predicted to give the cheapest mutation. If there is only 1 possible value, then that is the new number of state particles

4.3 Replacing the state particle set

Our final contribution (denoted REPLACE) is a variation of the REWEIGHT scheme of Chopin et al. (2012). Both REWEIGHT and REPLACE consist of three steps. First, a particle filter (Algorithm 1) is run with the new number of state particles to obtain $\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x^*})$ and $\boldsymbol{x}_{1:d}^{1:N_x^*}$. Second, the parameter particle weights are reweighted using

$$w_d^n = W_d^n \cdot IW_d^n, \quad W_d^n = \frac{w_d^n}{\sum_{i=1}^{N_\theta} w_d^i},$$

where IW_d^n is the incremental weight for parameter particle n, $n = 1, \ldots, N_\theta$ at iteration d; then the previous likelihood estimate and set of state particles are discarded. Note that prior to this reweighting step, the parameter particles are evenly weighted as the adaptation of N_x is performed after the resampling step, i.e. $W_d^n = 1/N_\theta$, for $n = 1, \ldots, N_\theta$.

With the REWEIGHT method, the incremental weights for DA-SMC^2 are obtained by replacing $p(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d)$ with $\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x})$ to approximate the optimal backward kernel, giving

$$IW_d = \frac{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x^*})}{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x})};$$

see Sect. 3 for details. For DT-SMC^2, the incremental weights are

$$IW_d = \frac{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x^*})^{g_d}}{\widehat{p_{N_x}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x})^{g_d}}.$$

The REPLACE method uses a different approximation to the optimal backward kernel. For DA-SMC^2, instead of using $p(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d) \approx p_{N_x}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x})$, we use $p(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d) \approx p_{N_x^*}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x^*})$, which gives the backward kernel

$$\begin{aligned} L_d(\boldsymbol{x}_{1:d}^{1:N_x^*}, \boldsymbol{x}_{1:d}^{1:N_x}) &= \frac{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x})\psi(\boldsymbol{x}_{1:d}^{1:N_x})}{p(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d)} \\ &\approx \frac{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x})\psi(\boldsymbol{x}_{1:d}^{1:N_x})}{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x^*})}. \end{aligned}$$

Using this backward kernel, the incremental weights are

$$\begin{aligned} IW_d &= \frac{\pi_d\left(\boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x^*} \mid \boldsymbol{y}_{1:d}\right) L_d(\boldsymbol{x}_{1:d}^{1:N_x^*}, \boldsymbol{x}_{1:d}^{1:N_x})}{\pi_d\left(\boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x} \mid \boldsymbol{y}_{1:d}\right) \psi(\boldsymbol{x}_{1:d}^{1:N_x^*})} \\ &= \frac{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x^*}) L_d(\boldsymbol{x}_{1:d}^{1:N_x^*}, \boldsymbol{x}_{1:d}^{1:N_x})}{\widehat{p_{N_x}}(\boldsymbol{y}_{1:d} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:d}^{1:N_x})\psi(\boldsymbol{x}_{1:d}^{1:N_x})} = 1. \end{aligned}$$

Similarly, for DT-SMC^2, the approximation $p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d)^{g_d} \approx \widehat{p_{N_x^*}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x^*})^{g_d}$ gives the backward kernel

$$\begin{aligned} L_d(\boldsymbol{x}_{1:T}^{1:N_x^*}, \boldsymbol{x}_{1:T}^{1:N_x}) &= \frac{\widehat{p_{N_x}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x})^{g_d}\psi(\boldsymbol{x}_{1:T}^{1:N_x})}{p(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d)^{g_d}} \\ &\approx \frac{\widehat{p_{N_x}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x})^{g_d}\psi(\boldsymbol{x}_{1:T}^{1:N_x})}{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x^*})^{g_d}}. \end{aligned}$$

leading to incremental weights

$$\begin{aligned} IW_d &= \frac{\pi_d\left(\boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x^*} \mid \boldsymbol{y}_{1:T}\right) L_d(\boldsymbol{x}_{1:T}^{1:N_x^*}, \boldsymbol{x}_{1:T}^{1:N_x})}{\pi_d\left(\boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x} \mid \boldsymbol{y}_{1:T}\right) \psi(\boldsymbol{x}_{1:T}^{1:N_x^*})} \\ &= \frac{\widehat{p_{N_x^*}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x^*})^{g_d} L_d(\boldsymbol{x}_{1:T}^{1:N_x^*}, \boldsymbol{x}_{1:T}^{1:N_x})}{\widehat{p_{N_x}}(\boldsymbol{y}_{1:T} \mid \boldsymbol{\theta}_d, \boldsymbol{x}_{1:T}^{1:N_x})^{g_d}\psi(\boldsymbol{x}_{1:T}^{1:N_x})} = 1. \end{aligned}$$

Since the incremental weights reduce to 1, the REPLACE approach introduces no extra variability in the parameter particle weights. Hence, REPLACE leads to less variability in the mutation step compared to the REWEIGHT method of Chopin et al. (2012), i.e. the parameter particles remain evenly weighted throughout the mutation step. We also find that it is generally faster than the REINIT method of Duan and Fulop (2014).

4.4 Practical considerations

The framework introduced in this section has a number of advantages over the existing methods. Most notably, the adaptation of R is automated, the stage 2 options (RESCALE-STD, NOVEL-VAR and RESCALE-ESJD) are less sensitive to variability in the estimated variance of the log-likelihood estimator, and the parameter particle weights are unchanged by adapting N_x.

Two tuning parameters remain to be specified for this method: the target ESJD ($\text{ESJD}_{\text{target}}$) and the number of samples to use when estimating the variance of the log-likelihood estimator (k). Our numerical experiments in Sect. 5 use $\text{ESJD}_{\text{target}} = 6$ and $k = 100$, which both give reasonable empirical results. The target ESJD has little effect on the value of N_x, due to the structure of the updates described in Sect. 4.2, but it directly controls R. Likewise, k controls the variability of $\widehat{\sigma_{N_x}}^2$. Recall that $\widehat{\sigma_{N_x}}^2$ is the estimated variance of the log-likelihood estimator with N_x state particles evaluated at the mean of the current set of parameter particles ($\bar{\boldsymbol{\theta}}_d$). Ideally, the value of k should change with N_x and $\bar{\boldsymbol{\theta}}_d$; however, it is not obvious how to do this. In general, we find that if $\sigma^2_{N_x} \approx \widehat{\sigma_{N_x}}^2$ is high, then the variance of $\widehat{\sigma_{N_x}}^2$ also tends to be high.

Determining optimal values of $\text{ESJD}_{\text{target}}$ and k is beyond the scope of this paper, but a general recommendation is to follow Salomone et al. (2018) and set $\text{ESJD}_{\text{target}}$ to the weighted average of the Mahalanobis distance between the parameter particles immediately before the resampling step. We also recommend choosing k such that the variance of $\widehat{\sigma_{N_x}}^2$ is low (< 0.1) when $\widehat{\sigma_{N_x}}^2 \approx 1$, i.e. the estimate of $\widehat{\sigma_{N_x}}^2$ should have low variance when it is around the target value. This value of k may be difficult to obtain, but again, we find that $k = 100$ gives reasonable performance across all the examples in Sect. 5. To mitigate the effect of a highly variable $\widehat{\sigma_{N_x}}^2$, it is also helpful to set a lower bound on the value of N_x, as well as an upper bound if a sensible one is known. An upper bound is also useful to restrict the amount of computational resources that is used by the algorithm.

Another advantage of our approach is that N_x can also be reduced. In general, we would expect N_x to increase at each iteration, based on the results $N_x = \mathcal{O}(t)$ (Andrieu et al. 2010) and $\text{var}(g_d \log \widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}))) = g_d^2 \text{var}(\log \widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta})))$. The former relates to DA-SMC^2 and suggests that N_x should increase as the length of the time series increases. The second result relates to DT-SMC^2. In this case, to obtain $g_d^2 \text{var}(\log \widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}))) = 1$, $\text{var}(\log \widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta})))$ must decrease (i.e. N_x must increase) as g_d increases. If the value of N_x is too high however, e.g. due to variability in the adaptation step at a previous iteration or if the initial value is higher than necessary, it is possible for N_x to decrease in subsequent iterations. Note that it is not feasible to allow N_x to decrease when using DOUBLE or REINIT.

5 Examples

5.1 Implementation

The methods are evaluated on a simple Brownian motion model, the one-factor stochastic volatility (SV) model in Chopin et al. (2012), and two ecological models: the theta-logistic model (Peters et al. 2010; Drovandi et al. 2022) and the noisy Ricker model (Fasiolo et al. 2016).

The code is implemented in MATLAB and is available at https://github.com/imkebotha/adaptive-exact-approximate-smc. The likelihood estimates are obtained using the bootstrap particle filter (Algorithm 1) with adaptive multinomial resampling, i.e. resampling is done whenever the effective sample size (ESS) drops below $N_x/2$. The results for all models, except for the Ricker model, are calculated from 50 independent runs, each with $N_\theta = 1000$ parameter samples. Due to time and computational constraints, the Ricker model results are based on 20 independent runs, each with $N_\theta = 400$ parameter samples.

For DT-SMC^2, the temperatures are set adaptively using the bisection method (Jasra et al. 2010) to aim for an ESS of $0.6 \cdot N_\theta$. Similarly, the resample-move step is run for DA-SMC^2 if the ESS falls below $0.6 \cdot N_\theta$. As discussed in Sect. 4.4, a target ESJD of 6 is used and the sample variance $\widehat{\sigma_{N_x}}^2$ for RESCALE-VAR, RESCALE-STD, NOVEL-VAR, and NOVEL-ESJD is calculated using $k = 100$ log-likelihood estimates. For all methods except REINIT and DOUBLE, we also trigger the adaptation whenever $\widehat{\text{ESJD}}_{t-1} > 2 \cdot \widehat{\text{ESJD}}_{\text{target}}$—this allows the algorithm to recover if the values of N_x and/or R are set too high at any given iteration, which may occur e.g. with DA-SMC^2 if there are outliers in the data. When the REINIT method is used, a mixture of three Gaussians is fit to the current sample when reinitialising the algorithm.

The methods are compared based on the mean squared error (MSE) of the posterior mean averaged over the parameters, where the ground truth is taken as the posterior mean from a PMMH chain of length 1 million. As the gold standard (GS), DT-SMC^2 and DA-SMC^2 are also run for each model with a fixed number of state particles, while still adapting R. For each of these runs, the number of state particles is tuned such that $\widehat{\sigma_{N_x}}^2 \approx 1$ for the full dataset, and the extra tuning time is not included in the results.

We use the MSE and the total number of log-likelihood evaluations (denoted TLL) of a given method as a measure of its accuracy and computational cost respectively. Note that each time the particle filter is run for a particular parameter particle, TLL is incremented by $N_x \times t$, where t is the current number of observations. The MSE multiplied by the TLL of a particular method gives its overall efficiency. Scores for the accuracy, computational cost and overall efficiency of a given method relative to the gold standard are calculated as

$$Z_{\text{method,MSE}} := \frac{\text{MSE}_{\text{GS}}}{\text{MSE}_{\text{method}}}, \quad Z_{\text{method,TLL}} := \frac{\text{TLL}_{\text{GS}}}{\text{TLL}_{\text{method}}},$$
$$Z_{\text{method}} := Z_{\text{method,MSE}} \times Z_{\text{method,TLL}}.$$

Higher values are better.

The adaptive mutation step in Algorithm 4 is used for all methods except NOVEL- ESJD, which uses the adaptive mutation step in Algorithm 5. The options for stage 2 are DOUBLE, RESCALE- VAR, RESCALE- STD, NOVEL- VAR and NOVEL- ESJD. Likewise, the options for stage 3 are REWEIGHT, REINIT, and our novel method REPLACE. Since the aim of the NOVEL- VAR method is to regularly increase the number of state particles throughout the iterations, the combination NOVEL- VAR with REINIT is not tested. Similarly, due to the number of times N_x is updated when using NOVEL- ESJD, only the combination NOVEL- ESJD with REPLACE is tested. For all combinations (excluding DOUBLE and REINIT), we allow the number of state particles to decrease. Due to computational constraints, we also cap the number of state particles at 5 times the number of state particles used for the the gold standard method. Note that the DOUBLE method cannot decrease N_x, and REINIT assumes increasing N_x throughout the iterations as the entire algorithm is reinitialised whenever N_x is updated.

To compare the different stage 2 methods, we also plot the evolution of N_x for each example. Recall that $N_x = \mathcal{O}(t)$ for DA-SMC^2 and $\text{var}(\log \widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta})^{g_d})) = g^2 \cdot \text{var}(\log \widehat{p_{N_x}}(\boldsymbol{y} \mid \boldsymbol{\theta}))$ for DT-SMC^2. Based on these two results, a roughly linear increase in N_x is desired—linear in time for DA-SMC^2 and linear in g^2 for DT-SMC^2. Section A of the Appendix shows marginal posterior density plots. Section B in the Appendix has extra results for the stochastic volatility model with $N_\theta = 100$ and $N_\theta = 500$, to test the methods with fewer parameter particles. We find that the variability of the adaptation of N_x increases with lower values of N_θ, which affects Z_{MSE} in particular. Based on the results in Section B, we recommend setting N_θ as high as possible subject to the available computational budget. There is also some evidence to suggest that higher values of N_x in the earlier iterations of DA-SMC^2 may be beneficial.

5.2 Brownian motion model

The first example is a stochastic differential equation with constant drift and diffusion coefficients,

$$dX_t = \left(\beta - \frac{\gamma^2}{2}\right) dt + \gamma dB_t,$$

where B_t is a standard Brownian motion process (Øksendal 2003, p. 44). The observation and transition densities are

$$g(y_t \mid x_t, \boldsymbol{\theta}) = \mathcal{N}(x_t, \sigma^2),$$
$$f(x_t \mid x_{t-1}, \boldsymbol{\theta}) = \mathcal{N}\left(x_{t-1} + \beta - \frac{\gamma^2}{2}, \gamma^2\right).$$

One hundred observations are generated from this model using $\boldsymbol{\theta} := (x_0, \beta, \gamma, \sigma) = (1, 1.2, 1.5, 1)$ and the priors assigned are $\mathcal{N}(x_0 \mid 3, 5^2)$, $\mathcal{N}(\beta \mid 2, 5^2)$, Half-Normal$(\gamma \mid 2^2)$, and Half-Normal$(\sigma \mid 2^2)$, respectively.

Results for all stage 2 and stage 3 combinations are obtained for initial N_x values of 10 and 100. The variance of the log-likelihood estimator is around 95 for $N_x = 10$ and around 2.7 for $N_x = 100$. The gold standard method is run with 240 state particles.

Table 2 shows the scores averaged over the two initial values of N_x for the three stage 3 options (REWEIGHT, REINIT and REPLACE). Note that these scores are relative to REWEIGHT instead of the gold standard. Apart from DT-SMC^2 with DOUBLE—where REINIT is faster than REPLACE—REPLACE consistently outperforms REWEIGHT and REINIT in terms of statistical and computational efficiency. Interestingly, REINIT generally outperforms REWEIGHT with RESCALE- STD and RESCALE- VAR, but not with DOUBLE. The performance of REINIT greatly depends on the number of times the algorithm is reinitialised and the final number of state particles, and this is generally reflected in the computation time.

Tables 3 and 4 show the scores relative to the gold standard for all the REPLACE combinations. NOVEL- ESJD has the best overall score followed by NOVEL- VAR for DT-SMC^2, and RESCALE- VAR for DA-SMC^2. DOUBLE performs well on DT-SMC^2, but poorly on DA-SMC^2—it has good statistical efficiency, but is much slower than the other methods. Interestingly, the computational efficiency is generally higher for the adaptive methods than for the gold standard, but their accuracy for DA-SMC^2 is generally lower. This may be due to high variability in the variance of the log-likelihood estimator and the mean of the parameter particles during the initial iterations of DA-SMC^2. Since fewer observations are used to estimate the likelihood in these early iterations ($t < T$), the mean of the parameter particles can change drastically from one iteration to the next, leading to similarly drastic changes in the sample variance of the log-likelihood estimator.

Figure 1 shows the evolution of N_x for REPLACE and an initial N_x of 10. Based on these plots, DOUBLE, NOVEL- VAR and NOVEL- ESJD have the most efficient adaptation for DT-SMC^2, and NOVEL- ESJD has the most efficient adaptation for DA-SMC^2, which corresponds with the results for Z_{TLL} and Z in Tables 3 and 4.

Table 2 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for the stage 3 options for the Brownian motion model—higher values are preferred

Method		DT-SMC^2			DA-SMC^2		
		Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
DOUBLE	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	REINIT	0.61	2.68	1.61	0.11	0.69	0.06
DOUBLE	REPLACE	1.18	1.17	1.46	1.86	1.68	2.99
RESCALE- VAR	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
RESCALE- VAR	REINIT	3.03	1.06	3.64	1.65	1.02	1.68
RESCALE- VAR	REPLACE	2.97	4.76	17.46	10.59	1.91	19.49
RESCALE- STD	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
RESCALE- STD	REINIT	5.56	1.93	11.34	1.04	1.30	1.47
RESCALE- STD	REPLACE	6.83	5.45	35.28	5.10	1.61	8.66

The results are averaged over the two starting values of N_x and are relative to the REWEIGHT method

Table 3 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for DT-SMC^2 for the Brownian motion model using the REPLACE method—higher values are preferred

Method	DT-SMC^2					
Initial N_x	10			100		
	Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
GOLD STANDARD	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	4.31	3.24	20.00	7.07	0.57	5.93
RESCALE- VAR	3.32	1.21	6.24	3.68	1.21	6.71
RESCALE- STD	4.82	2.47	18.30	4.96	1.44	10.96
NOVEL- VAR	4.21	3.26	21.01	3.89	2.43	14.41
NOVEL- ESJD	1.95	8.75	26.34	3.58	2.42	13.16

The gold standard refers to SMC^2 with a fixed number of state particles

Table 4 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for DA-SMC^2 for the Brownian motion model using the REPLACE method—higher values are preferred

Method	DA-SMC^2					
Initial N_x	10			100		
	Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
GOLD STANDARD	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	1.13	0.37	0.53	1.42	0.12	0.17
RESCALE- VAR	1.11	2.09	1.93	0.68	2.10	1.53
RESCALE- STD	0.50	2.52	1.36	0.58	2.27	1.44
NOVEL- VAR	0.76	1.93	1.47	0.73	1.68	1.13
NOVEL- ESJD	0.74	2.95	2.49	0.71	2.75	1.95

The gold standard refers to SMC^2 with a fixed number of state particles

5.3 Stochastic volatility model

Our second example is the one-factor stochastic volatility model used in Chopin et al. (2012),

$$y_t \sim \mathcal{N}(\mu + \beta v_t, v_t),$$

$$z_t = \exp(-\lambda) z_{t-1} + \sum_{j=1}^{k} \exp(-\lambda(t - c_j)) e_j,$$

$$z_0 \sim \text{Gamma}(\xi^2/\omega^2, \xi/\omega^2)$$

$$v_t = \frac{1}{\lambda}\left[z_{t-1} - z_t + \sum_{j=1}^{k} e_j\right], \quad x_t = \{v_t, z_t\},$$

$$k \sim \text{Poisson}(\lambda\xi^2/\omega^2), \quad c_{1:k} \overset{\text{iid}}{\sim} \text{Uniform}(t-1, t),$$

$$e_{1:k} \overset{\text{iid}}{\sim} \text{Exponential}(\xi/\omega^2).$$

The transition density of this model cannot be evaluated point-wise, but it can be simulated from.

We use a synthetic dataset with 200 observations, which is generated using $\boldsymbol{\theta} := (\xi, \omega^2, \lambda, \beta, \mu) = (4, 4, 0.5, 5, 0)$. The priors are Exponential$(\xi \mid 0.2)$, Exponential$(\omega^2 \mid 0.2)$, Exponential$(\lambda \mid 1)$, $\mathcal{N}(\beta \mid 0, 2)$ and $\mathcal{N}(\mu \mid 0, 2)$.

Results for all stage 2 and stage 3 combinations are obtained for initial N_x values of 300 and 600. The variance of the log-likelihood estimator is around 7 for 300 state particles and around 3 for 600 state particles. The gold standard method is run with 1650 state particles.

Table 5 shows the scores for the three stage 3 options, relative to REWEIGHT and averaged over the two initial N_x values. REPLACE consistently outperforms REWEIGHT and REINIT in terms of overall efficiency.

Tables 6 and 7 show the scores for all the REPLACE combinations. All methods perform similarly for this model. In terms of accuracy (measured by the MSE), the optimal variance of the log-likelihood estimator seems to be smaller for this model than for the others. However, the efficiency of a smaller variance coupled with the increased computation time is fairly similar to the efficiency of a larger variance with cheaper computation. In this example, NOVEL- ESJD has the highest MSE, but the lowest computation time.

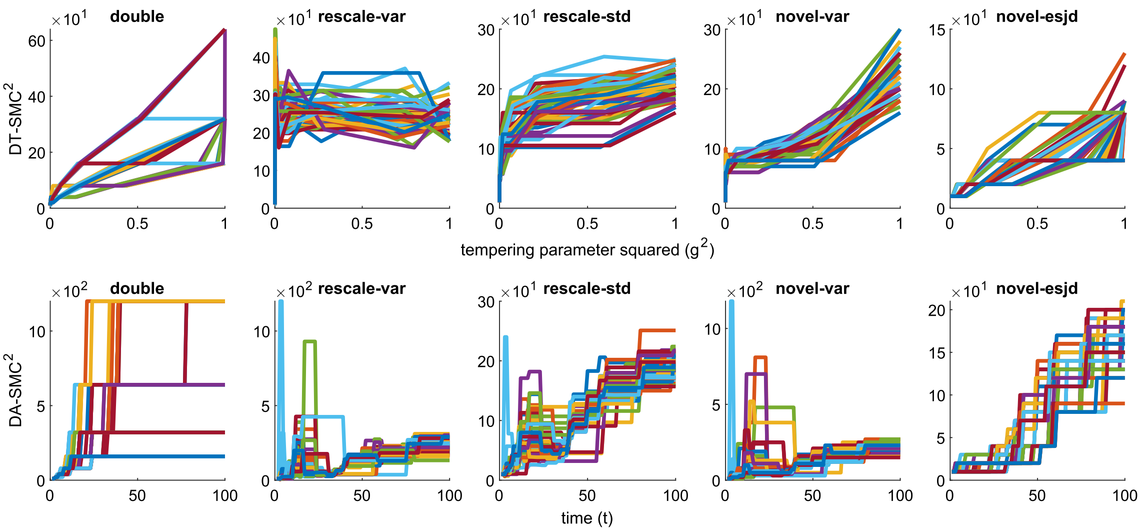


Fig. 1 Evolution of N_x for REPLACE and a low initial N_x for the Brownian motion model. Each coloured line represents an independent run of the given method. The maximum N_x is 1200

Table 5 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for the stage 3 options for the stochastic volatility model—higher values are preferred

Method		DT-SMC^2			DA-SMC^2		
		Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
DOUBLE	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	REINIT	1.07	0.88	0.83	0.71	0.41	0.17
DOUBLE	REPLACE	1.38	1.15	1.48	5.09	1.10	4.31
RESCALE- VAR	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
RESCALE- VAR	REINIT	4.41	1.49	7.06	0.78	0.65	0.42
RESCALE- VAR	REPLACE	2.92	5.40	17.24	5.06	1.07	4.60
RESCALE- STD	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
RESCALE- STD	REINIT	7.33	2.07	16.21	0.26	0.44	0.13
RESCALE- STD	REPLACE	4.49	5.07	24.12	1.93	1.04	1.91

The results are averaged over the two starting values of N_x and are relative to the REWEIGHT method

Figure 2 shows the evolution of N_x for REPLACE and an initial N_x of 300. Based on these plots, DOUBLE and NOVEL-ESJD have the most efficient adaptation for DT-SMC^2, and all methods except DOUBLE have good results for DA-SMC^2. These methods correspond to those with the quickest run time (lowest TLL), but not to the ones with the best overall efficiency.

5.4 Theta-logistic model

The theta-logistic ecological model (Peters et al. 2010) is

$$g(y_t \mid x_t, \boldsymbol{\theta}) = \mathcal{N}(y_t \mid a \cdot (x_t), \sigma^2),$$
$$x_{t+1} = x_t + \beta_0 + \beta_1 \exp(\beta_2 x_t) + z_t, \quad z_t \sim \mathcal{N}(0, \gamma^2).$$

We fit the model to the first 100 observations of female nutria populations measured at monthly intervals (Peters et al. 2010; Drovandi et al. 2022), using the priors $\mathcal{N}(\beta_0 \mid 0, 1)$, $\mathcal{N}(\beta_1 \mid 0, 1)$, $\mathcal{N}(\beta_2 \mid 0, 1)$, Half-Normal$(\exp(x_0) \mid 1000^2)$, Exponential$(\gamma \mid 1)$, Exponential$(\sigma \mid 1)$ and $\mathcal{N}(a \mid 1, 0.5^2)$.

Scores for the accuracy, computational cost and overall efficiency are obtained for initial N_x values of 700 and 2400. The variance of the log-likelihood estimator is around 40 for 700 state particles and around 3 for 2400 state particles. The gold standard method is run with 4600 state particles. Due to time constraints, results for the DOUBLE method with REWEIGHT and initial $N_x = 700$ are not available for DA-SMC^2.

Table 8 shows the scores for the three stage 3 options, averaged over the initial N_x values and relative to REWEIGHT. Except for DOUBLE with DA-SMC^2, both REINIT and REPLACE outperform REWEIGHT, but the results for REINIT and REPLACE are mixed. The performance of REINIT greatly depends on the number of times the adaptation is triggered. On average, the algorithm is reinitialised fewer times for RESCALE- STD for this example than for the others.

Table 6 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for DT-SMC^2 for the stochastic volatility model using the REPLACE method—higher values are preferred

Method	DT-SMC^2					
Initial N_x	300			600		
	Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
GOLD STANDARD	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	1.16	2.04	2.82	1.63	1.02	1.78
RESCALE- VAR	1.73	0.86	1.52	1.68	0.79	1.40
RESCALE- STD	1.26	1.66	2.20	1.35	1.27	1.75
NOVEL- VAR	1.16	1.89	2.23	1.15	1.59	1.88
NOVEL- ESJD	0.52	3.82	2.03	0.82	2.09	1.73

The gold standard refers to SMC^2 with a fixed number of state particles

Table 7 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for DA-SMC^2 for the stochastic volatility model using the REPLACE method—higher values are preferred

Method	DA-SMC^2					
Initial N_x	300			600		
	Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
GOLD STANDARD	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	1.43	0.51	0.74	1.53	0.37	0.56
RESCALE- VAR	0.80	1.34	1.06	0.71	1.33	0.96
RESCALE- STD	0.77	1.40	1.08	0.63	1.41	0.91
NOVEL- VAR	0.75	1.38	1.05	0.91	1.38	1.28
NOVEL- ESJD	0.63	1.35	0.89	0.67	1.31	0.88

The gold standard refers to SMC^2 with a fixed number of state particles

Tables 9 and 10 show the scores for all the REPLACE combinations relative to the gold standard. In this example, NOVEL- ESJD outperforms all other methods, followed by NOVEL- VAR and RESCALE- VAR. Unlike the previous examples, DOUBLE and RESCALE- STD perform poorly here. The

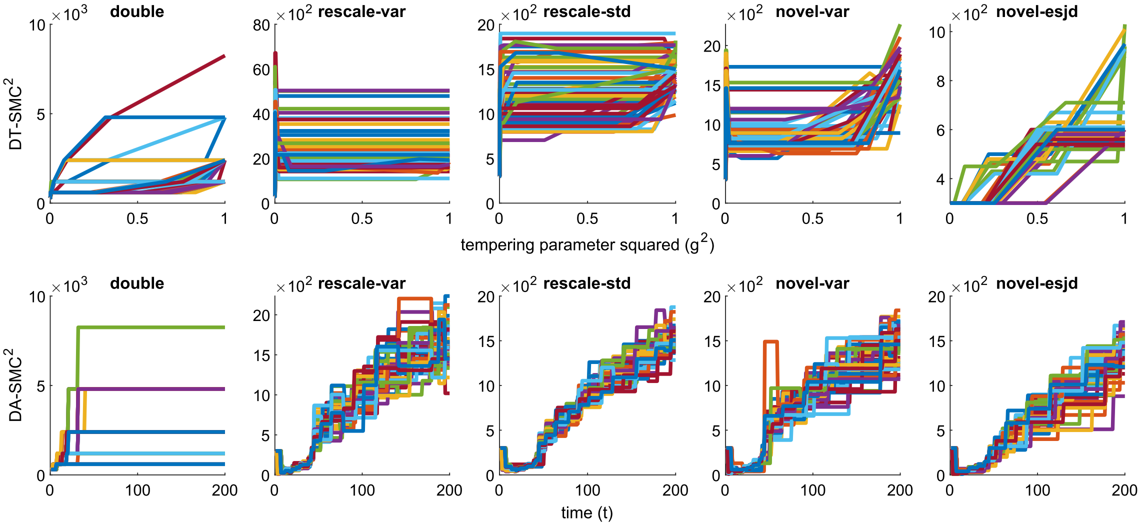


Fig. 2 Evolution of N_x for REPLACE and a low initial N_x for the stochastic volatility model. Each coloured line represents an independent run. The maximum N_x is 8250

Table 8 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for the stage 3 options for the theta-logistic model—higher values are preferred

Method		DT-SMC^2			DA-SMC^2		
		Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
DOUBLE	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	REINIT	0.18	7.89	1.31	0.09	1.80	0.16
DOUBLE	REPLACE	1.18	0.94	1.11	0.85	1.09	0.89
RESCALE- VAR	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
RESCALE- VAR	REINIT	0.98	6.84	7.28	0.99	1.78	1.67
RESCALE- VAR	REPLACE	1.02	2.41	1.91	0.71	3.46	2.64
RESCALE- STD	REWEIGHT	1.00	1.00	1.00	1.00	1.00	1.00
RESCALE- STD	REINIT	0.99	4.14	4.24	0.76	2.42	1.78
RESCALE- STD	REPLACE	1.36	1.75	3.75	0.69	3.73	2.51

The results are averaged over the two starting values of N_x and are relative to the REWEIGHT method

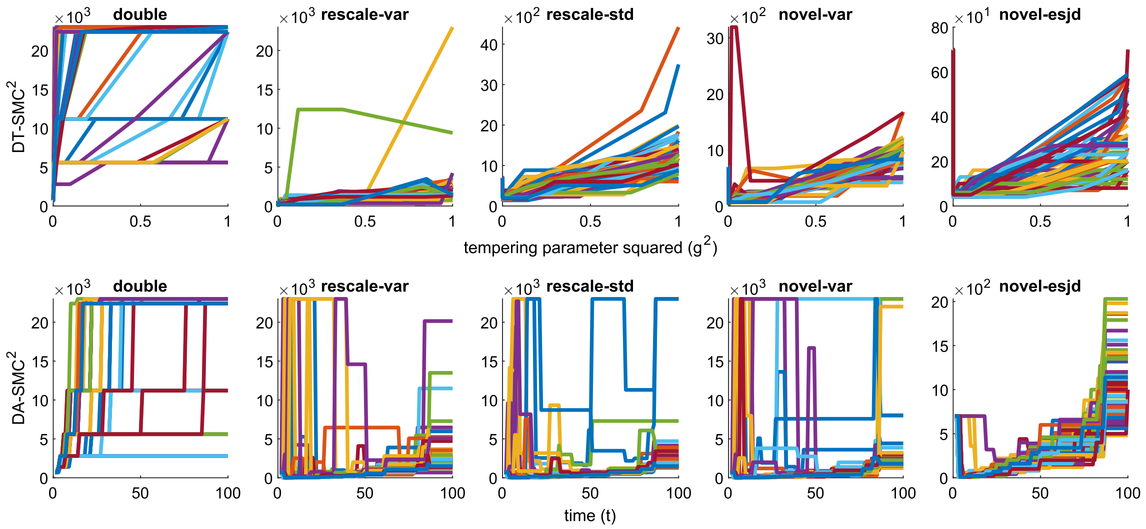


Fig. 3 Evolution of N_x for REPLACE and a low initial N_x for the theta-logistic model. Each coloured line represents an independent run. The maximum N_x is 23000

Table 9 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for DT-SMC^2 for the theta-logistic model using the REPLACE method—higher values are preferred

Method	DT-SMC^2					
Initial N_x	700			2400		
	Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
GOLD STANDARD	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	1.35	0.38	0.52	1.32	0.28	0.37
RESCALE- VAR	0.16	5.32	1.14	0.16	5.43	0.89
RESCALE- STD	0.13	11.23	1.49	0.14	4.39	0.76
NOVEL- VAR	0.09	20.00	1.87	0.09	9.50	1.00
NOVEL- ESJD	0.06	34.78	2.11	0.06	19.37	1.14

The gold standard refers to SMC^2 with a fixed number of state particles

gold standard and DOUBLE have the best MSE for this example, but the worst computation time. The remaining methods have a poor MSE, which is mostly due to the parameter σ as Fig. 7 in Section A of the Appendix shows. The gold standard is the only method that achieves a good result for σ.

Figure 3 shows the evolution of N_x for REPLACE and an initial N_x of 700. NOVEL- ESJD seem to have the least variable evolution for both DT-SMC^2 and DA-SMC^2 compared to the other methods. Again, this is reflected in the values of Z_{TLL}, particularly in Tables 9 and 10.

5.5 Noisy Ricker model

Our final example is the noisy Ricker population model (Fasiolo et al. 2016),

$$g(y_t \mid x_t, \boldsymbol{\theta}) = \text{Poisson}(y_t \mid \phi x_t),$$
$$x_{t+1} = r \cdot x_t \exp(-x_t + z_t), \quad z_t \sim \mathcal{N}(0, \sigma^2).$$

Table 10 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for DA-SMC^2 for the theta-logistic model using the REPLACE method—higher values are preferred

Method	DA-SMC^2					
Initial N_x	700			2400		
	Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
GOLD STANDARD	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	1.33	0.22	0.33	1.45	0.15	0.30
RESCALE- VAR	0.25	2.17	1.09	0.24	1.86	1.00
RESCALE- STD	0.17	2.48	0.31	0.19	2.37	0.83
NOVEL- VAR	0.24	1.87	0.67	0.21	2.18	1.04
NOVEL- ESJD	0.13	13.42	2.05	0.12	12.38	1.76

The gold standard refers to SMC^2 with a fixed number of state particles

The Ricker model, and its variants, is typically used to represent highly non-linear or near-chaotic ecological systems, e.g. the population dynamics of sheep blowflies (Fasiolo et al. 2016). Fasiolo et al. (2016) show that the likelihood function of the noisy Ricker model exhibits extreme multimodality when the process noise is low, making it difficult to estimate the model.

We draw 700 observations using $\boldsymbol{\theta} := (\log(\phi), \log(r), \log(\sigma)) = (\log(10), \log(44.7), \log(0.6))$. Following Fasiolo et al. (2016), we assign uniform priors to the log-parameters, $\mathcal{U}(\log(\phi) \mid 1.61, 3)$, $\mathcal{U}(\log(r) \mid 2, 5)$ and $\mathcal{U}(\log(\sigma) \mid -1.8, 1)$, respectively.

Scores for the accuracy, computational cost and overall efficiency are obtained for initial N_x values of 1000 and 20000. The variance of the log-likelihood estimator is around 13 for 1000 state particles and around 2.3 for 20000 state particles. The gold standard method is run with 90000 state

Table 11 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for DT-SMC^2 for the noisy Ricker model using the REPLACE method—higher values are preferred

Method	DT-SMC^2					
Initial N_x	1000			20000		
	Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
GOLD STANDARD	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	0.26	12.76	3.59	–	–	–
RESCALE- VAR	0.45	4.17	2.10	0.77	3.34	2.82
RESCALE- STD	0.33	12.79	4.62	0.54	4.28	2.40
NOVEL- VAR	0.38	10.76	4.03	0.37	7.16	2.90
NOVEL- ESJD	0.12	46.19	5.63	0.24	10.51	2.65

The gold standard refers to SMC^2 with a fixed number of state particles

Table 12 Scores for the accuracy (Z_{MSE}), computational cost (Z_{TLL}) and overall efficiency (Z) for DA-SMC^2 for the noisy Ricker model using the REPLACE method—higher values are preferred

Method	DA-SMC^2					
Initial N_x	1000			20000		
	Z_{MSE}	Z_{TLL}	Z	Z_{MSE}	Z_{TLL}	Z
GOLD STANDARD	1.00	1.00	1.00	1.00	1.00	1.00
DOUBLE	–	–	–	–	–	–
RESCALE- VAR	0.56	2.04	1.24	0.78	2.11	1.82
RESCALE- STD	0.47	3.33	1.63	0.41	3.09	1.28
NOVEL- VAR	0.87	2.00	1.78	1.02	2.29	2.47
NOVEL- ESJD	0.32	6.17	2.16	0.43	5.38	2.46

The gold standard refers to SMC^2 with a fixed number of state particles

particles. Due to time constraints, the ground truth for the posterior mean is based on a PMMH chain of length 200000.

An experiment was stopped if its run time exceeded 9 days. Hence, a full comparison of the stage 3 options cannot be made. Of the experiments that finished, REPLACE had the best results in terms of overall efficiency. On average, REPLACE outperformed REINIT and REWEIGHT by at least a factor of 2. In a number of cases, the gold standard and REPLACE were the only methods to finish within the time frame. Tables 11 and 12 show the scores for the REPLACE combinations. NOVEL- VAR and NOVEL- ESJD have the best overall results across both DT-SMC^2 and DA-SMC^2 for this example, while RESCALE- STD and RESCALE- VAR perform similarly.

Figure 4 shows the evolution of N_x for REPLACE and an initial N_x of 1000. All methods show a fairly smooth increase in N_x over the iterations.

6 Discussion

We introduce an efficient SMC^2 algorithm which automatically updates the number of state particles throughout the algorithm. Of the methods used to select the new number of state particles, NOVEL- ESJD gives the most consistent results across all models, choice of initial N_x and between DT-SMC^2 and DA-SMC^2. This method uses the ESJD to determine which N_x from a set of candidate values will give the cheapest mutation—this value is selected as the new number of state particles. NOVEL- ESJD generally outperforms the other methods in terms of the computational and overall efficiency. A significant advantage of NOVEL- ESJD is that the adaptation of N_x is consistent across independent runs of the algorithm

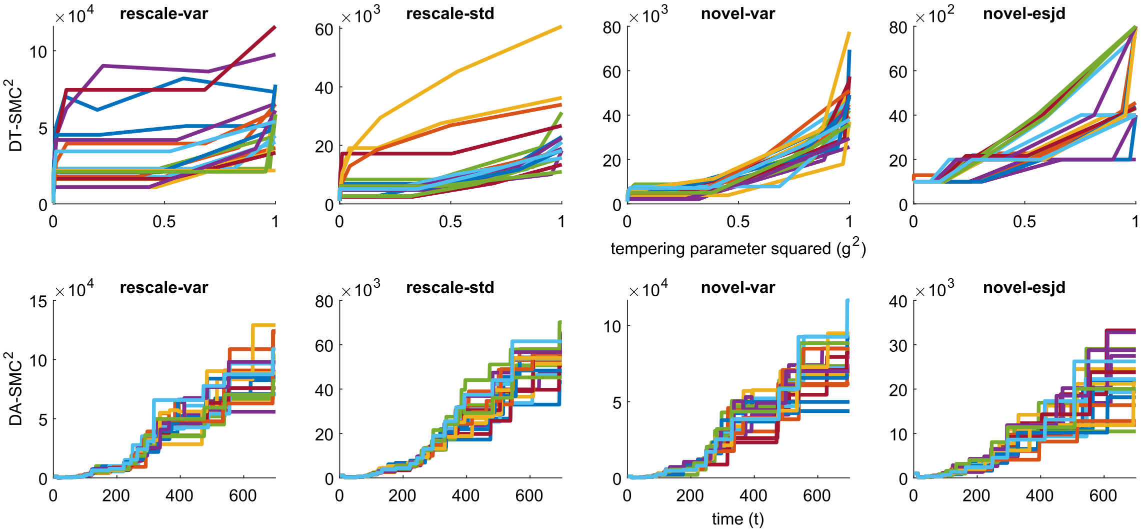


Fig. 4 Evolution of N_x for REPLACE and a low initial N_x for the Ricker model. Each coloured line represents an independent run. The maximum N_x is 450000

(i.e. when starting at different random seeds), substantially more so than the other methods.

Similarly, the REPLACE method typically shows great improvement over REWEIGHT and REINIT. REPLACE modifies the approximation to the optimal backward kernel used by REWEIGHT. This modification means that, unlike REWEIGHT, REPLACE leaves the parameter particle weights unchanged. We also find that REPLACE is generally more reliable than REINIT.

Our novel SMC^2 algorithm has three tuning parameters that must be set: the target ESJD for the mutation step, the number of log-likelihood evaluations for the variance estimation (k) and the initial number of state particles. Determining optimal values of the target ESJD and k is beyond the scope of this paper, but tuning strategies are discussed in Sect. 4.4. While any initial number of state particles can be used, a small value yields the most efficient results. Compared to the currently available methods, the new approach requires minimal tuning, gives consistent results and is straightforward to use with both data annealing and density tempering SMC^2. We also find that the adaptive methods generally outperform the gold standard, despite the latter being pre-tuned.

An interesting extension to the current work would be to assess the effect of the target ESJD, the target ESS and the target variance of the log-likelihood estimator when SMC^2 is used for model selection. Another area of future work is extending the method for application to mixed effects models (Botha et al. 2021); for these models, it may be possible to obtain significant gains in efficiency by allowing the number of state particles to (adaptively) vary between subjects. The new method can also be used as the proposal function within importance sampling squared (Tran et al. 2020).

One area of future work is to adapt the number of parameter particles, e.g. using the adaptive particle filters of Bhadra and Ionides (2016) and Elvira et al. (2017). Another area of future work is to adapt the number of parameter particles (N_θ) for a specific purpose, e.g. estimation of a particular parameter or subset of parameters. This may reduce the computational resources needed, and applies to SMC methods in general.

Acknowledgements Imke Botha was supported by an Australian Research Training Program Stipend and a QUT Centre for Data Science Top-Up Scholarship. Christopher Drovandi was supported by an Australian Research Council Discovery Project (DP200102101). We gratefully acknowledge the computational resources provided by QUT's High Performance Computing and Research Support Group (HPC).

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

A Marginal posterior plots

This section shows the marginal posterior density plots for the examples in Sects. 5.2–5.5. Figures 5, 6, 7 and 8 are the marginal posterior density plots for each example and method. Note that the results shown are for REPLACE using the combined samples from the independent runs, i.e. the

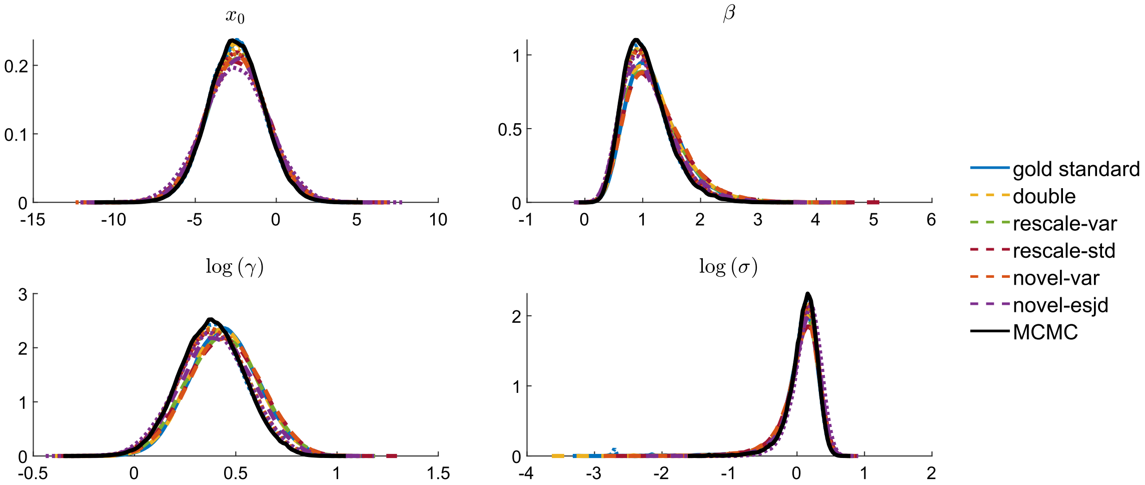


Fig. 5 Marginal posterior density plots for the Brownian motion model. Dashed lines are the DA-SMC^2 results and dotted lines of the same colour are the corresponding DT-SMC^2 results

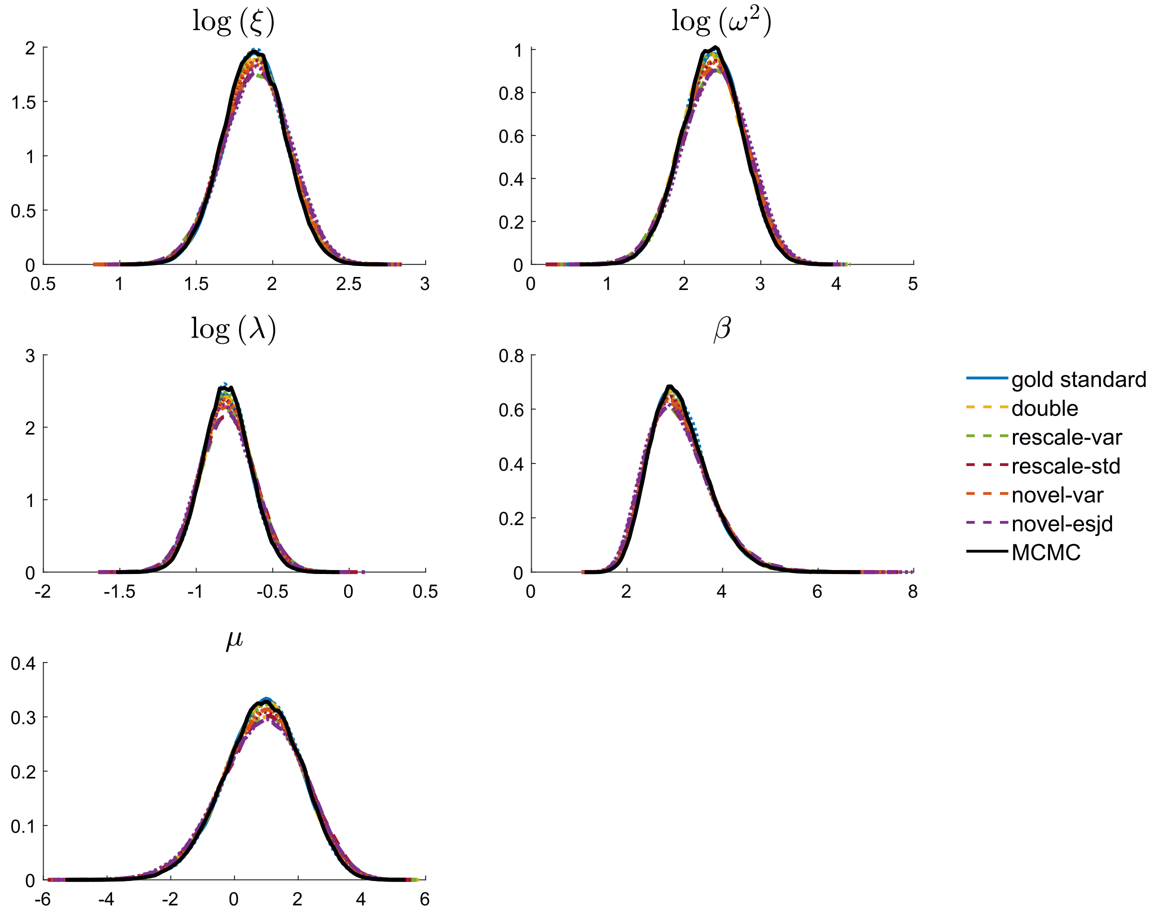


Fig. 6 Marginal posterior density plots for the stochastic volatility model. Dashed lines are DA-SMC^2 results and dotted lines of the same colour are the corresponding DT-SMC^2 results

marginal posteriors are based on 50 × 1000 samples for the Brownian motion, stochastic volatility and theta-logistic models and 20 × 400 samples for the Ricker model. The results shown are for a low initial N_x. The plots show that the marginal posterior densities are similar between the adaptive methods. The biggest difference in densities is between DT-SMC^2 and DA-SMC^2, not between the adaptive methods. Figures 5, 6 and 8 show marginal posteriors from SMC^2 that are very similar to the marginal posteriors from MCMC. Figure 7 shows similar marginal posteriors for the theta-logistic model from SMC^2 and MCMC for all of the parameters except for $\log(\sigma)$. This parameter corresponds to the log of the measurement error in the nutria population data (see Sect. 5.4 of the main paper). Here, the adaptive SMC^2 methods struggle to accurately capture the left tail of $\log(\sigma)$. SMC^2 with a higher, fixed number of state particles (the gold standard method) does not have the same issue, suggesting that the number of state particles is perhaps not adapted high enough in any of the methods for this example.

B Extra results for the stochastic volatility model

This section shows extra results for the stochastic volatility model. Figure 9 shows the difference in the scores for the three values of N_θ. Interestingly, Z_{TLL} is fairly constant for all methods, but Z_{MSE} is variable. The latter is generally at its highest when $N_\theta = 1000$ for DT-SMC^2, and when $N_\theta = 100$ for DA-SMC^2. Recall that Z_{MSE} is relative to the gold standard. DA-SMC^2 has the highest Z_{MSE} when $N_\theta = 100$, but also the highest Z_{TLL}. We find, on average,

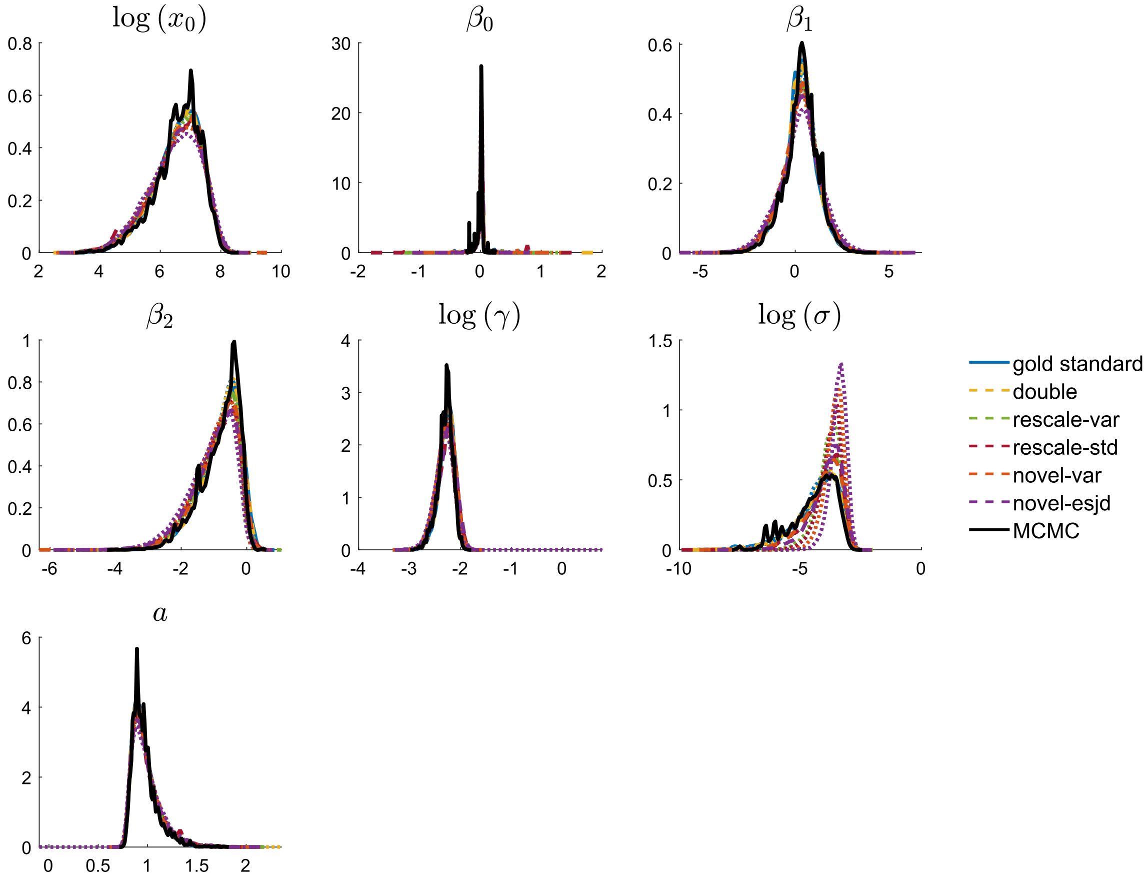
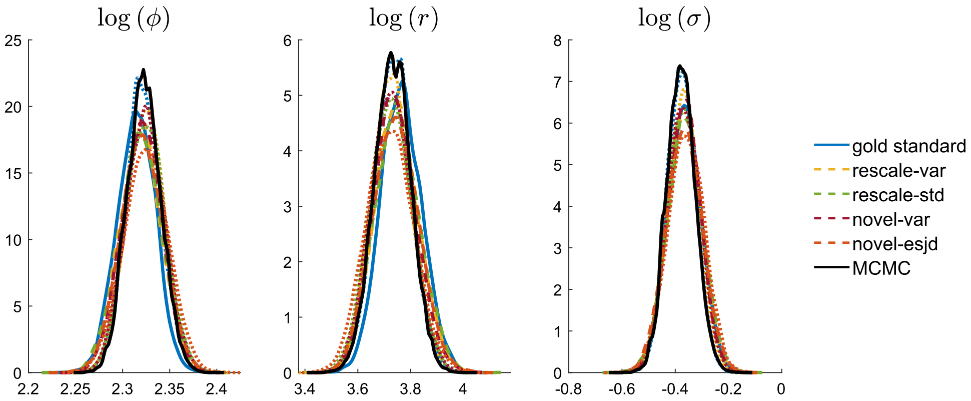


Fig. 7 Marginal posterior density plots for the theta-logistic model. Dashed lines are the DA-SMC^2 results and dotted lines of the same colour are the corresponding DT-SMC^2 results

Fig. 8 Marginal posterior density plots for the Ricker model. Dashed lines are the DA-SMC^2 results and dotted lines of the same colour are the corresponding DT-SMC^2 results



that the value of N_x is higher in the initial stages of DA-SMC^2 when N_θ is small compared to when N_θ is larger. This may be due to extra variability in the adaptation of N_x resulting from a small N_θ. However, these results indicate that a higher number of state particles may be beneficial for DA-SMC^2 at earlier iterations—something between DOUBLE and the other methods.

Figure 10 shows the (non-relative) MSE and computation cost (TLL) of the methods. For all methods, including the gold standard, there is a large improvement in the MSE when increasing N_θ from 100 to 500, but only a slight improvement when N_θ is increased from 500 to 1000. Based on the results shown in Figs. 9 and 10, we recommend setting N_θ as high as possible subject to the available computational budget.

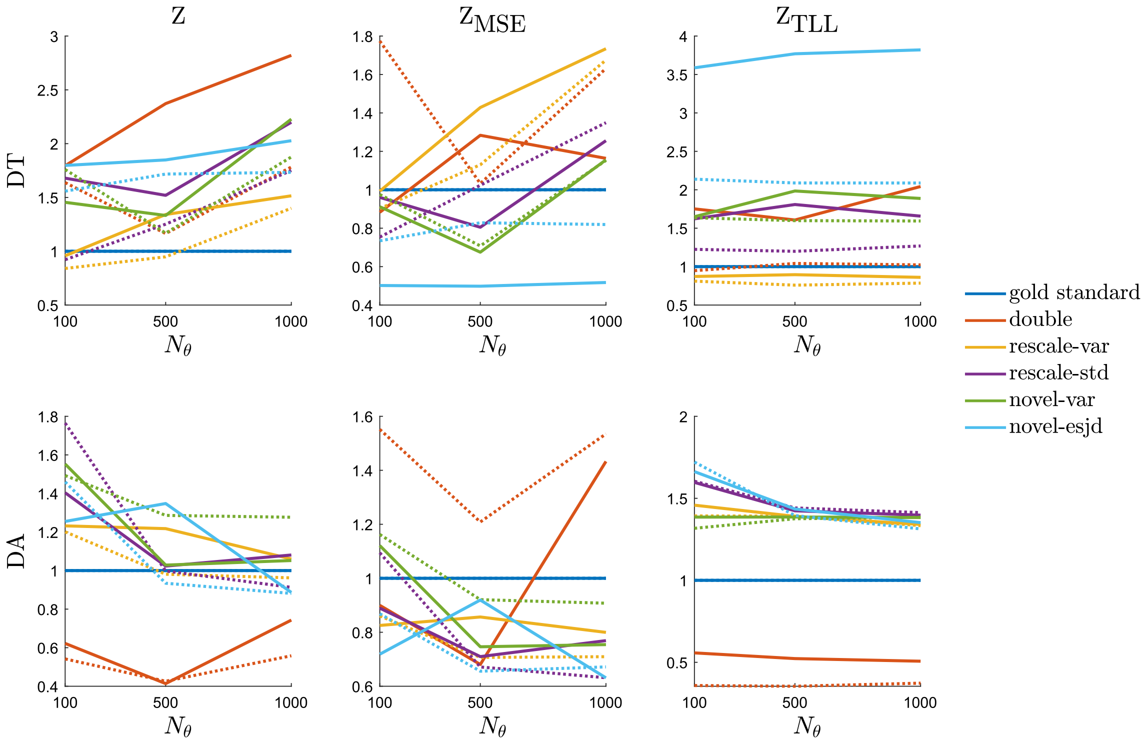


Fig. 9 Scores for the overall efficiency (Z), accuracy (Z_{MSE}) and computational cost (Z_{TLL}) for $N_\theta = 100, 500, 1000$ (left to right). The solid and dashed lines correspond to $N_x = 300$ and $N_x = 600$ respectively. The first row of plots gives the results for density tempering SMC^2, and the bottom row of plots gives the results for data annealing SMC^2. These results are relative to the gold standard (solid blue at $y = 1$), meaning that higher values are preferred

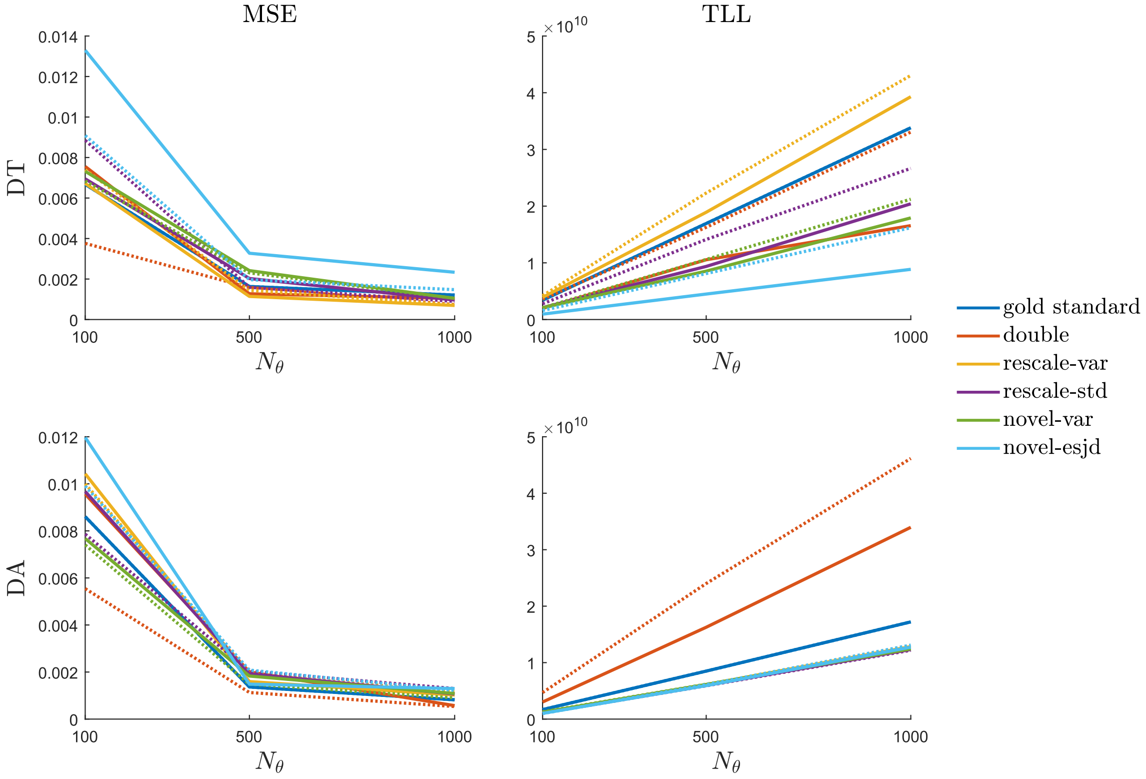


Fig. 10 Values for the error (MSE) and computation cost (TLL) for $N_\theta = 100, 500, 1000$. The solid and dashed lines correspond to $N_x = 300$ and $N_x = 600$ respectively. The first row of plots gives the results for density tempering SMC^2, and the bottom row of plots gives the results for data annealing SMC^2. The solid blue line represents the gold standard. Lower values are preferred

References

Andrieu, C., Doucet, A., Holenstein, R.: Particle Markov chain Monte Carlo methods. J. R. Stat. Soc.: Ser. B (Stat. Methodol.) **72**(3), 269–342 (2010)

Andrieu, C., Roberts, G.O.: The pseudo-marginal approach for efficient Monte Carlo computations. Ann. Stat. **37**(2), 697–725 (2009)

Bhadra, A., Ionides, E.L.: Adaptive particle allocation in iterated sequential Monte Carlo via approximating meta-models. Stat. Comput. **26**(1–2), 393–407 (2016)

Bon, J.J., Lee, A., Drovandi, C.: Accelerating sequential Monte Carlo with surrogate likelihoods. Stat. Comput. **31**(5), 1 (2021)

Botha, I., Kohn, R., Drovandi, C.: Particle methods for stochastic differential equation mixed effects models. Bayesian Anal. **16**(2), 1 (2021)

Cappé, O., Moulines, E., Rydén, T.: Inference in Hidden Markov Models. Springer, New York (2005)

Chopin, N.: A sequential particle filter method for static models. Biometrika **89**(3), 539–552 (2002)

Chopin, N., Jacob, P.E., Papaspiliopoulos, O.: SMC2: an efficient algorithm for sequential analysis of state space models. J. R. Stat. Soc.: Ser. B (Stat. Method.) **75**(3), 397–426 (2012)

Chopin, N., Ridgway, J., Gerber, M., Papaspiliopoulos, O.: Towards automatic calibration of the number of state particles within the SMC^2 algorithm. arXiv preprint arXiv:1506.00570 (2015)

Crisan, D., Míguez, J.: Uniform convergence over time of a nested particle filtering scheme for recursive parameter estimation in state-space Markov models. Adv. Appl. Probab. **49**(4), 1170–1200 (2017)

Crisan, D., Míguez, J.: Nested particle filters for online parameter estimation in discrete-time state-space Markov models. Bernoulli **24**(4A), 3039–3086 (2018)

Del Moral, P.: Feynman-Kac Formulae. Springer, New York (2004)

Del Moral, P., Doucet, A., Jasra, A.: Sequential Monte Carlo samplers. J. R. Stat. Soc.: Ser. B (Stat. Methodol.) **68**(3), 411–436 (2006)

Doucet, A., Pitt, M.K., Deligiannidis, G., Kohn, R.: Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. Biometrika **102**(2), 295–313 (2015)

Doucet, A., Pitt, M. K., and Kohn, R. (2012). Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. arXiv preprint arXiv:1210.1871v2

Drovandi, C., Everitt, R.G., Golightly, A., Prangle, D.: Ensemble MCMC: accelerating pseudo-marginal MCMC for state space models using the ensemble Kalman filter. Bayesian Anal **17**(1), 1 (2022)

Duan, J.-C., Fulop, A.: Density-tempered marginalized sequential Monte Carlo samplers. J. Bus. Econ. Stat. **33**(2), 192–202 (2014)

Elvira, V., Miguez, J., Djurić, P.M.: On the performance of particle filters with adaptive number of particles. Stat. Comput. **31**(6), 1 (2021)

Elvira, V., Miguez, J., Djurie, P.M.: Adapting the number of particles in sequential Monte Carlo methods through an online scheme for convergence assessment. IEEE Trans. Signal Process. **65**(7), 1781–1794 (2017)

Fasiolo, M., Pya, N., Wood, S.N.: A comparison of inferential methods for highly nonlinear state space models in ecology and epidemiology. Stat. Sci. **31**(1), 96–118 (2016)

Fearnhead, P., Taylor, B.M.: An adaptive sequential Monte Carlo sampler. Bayesian Anal. **8**(2), 411–438 (2013)

Fox, D.: Adapting the sample size in particle filters through KLD-sampling. Int. J. Robot. Res. **22**(12), 985–1003 (2003)

Gordon, N.J., Salmond, D.J., Smith, A.F.M.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEEE Proc. F Radar Signal Process. **140**(2), 107 (1993)

Green, P.J.: Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika **82**(4), 711–732 (1995)

Gunawan, D., Kohn, R., Tran, M.N.: Robust Particle Density Tempering for State Space Models. arXiv preprint arXiv:1805.00649 (2021)

Jacob, P.E., Murray, L.M., Rubenthaler, S.: Path storage in the particle filter. Stat. Comput. **25**(2), 487–496 (2015)

Jasra, A., Stephens, D.A., Doucet, A., Tsagaris, T.: Inference for Lévy–Driven Stochastic Volatility Models via Adaptive Sequential Monte Carlo. Scand. J. Stat. **38**(1), 1–22 (2010)

Lee, A., Whiteley, N.: Variance estimation in the particle filter. Biometrika **105**(3), 609–625 (2018)

Øksendal, B.: Stochastic differential equations: an introduction with applications. Springer, Berlin (2003)

Pasarica, C., Gelman, A.: Adaptively scaling the metropolis algorithm using expected squared jumped distance. Stat. Sin. **20**(1), 343–364 (2010)

Peters, G.W., Hosack, G.R., Hayes, K.R.: Ecological non-linear state space model selection via adaptive particle Markov chain Monte Carlo (AdPMCMC) (2010). arXiv preprints arXiv:1005.2238

Pitt, M.K., dos Santos Silva, R., Giordani, P., Kohn, R.: On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. J. Econom. **171**(2), 134–151 (2012)

Salomone, R., South, L.F., Drovandi, C.C., Kroese, D.P.: Unbiased and Consistent Nested Sampling via Sequential Monte Carlo (2018). arxiv preprint arXiv:1805.03924

Sherlock, C., Thiery, A.H., Roberts, G.O., Rosenthal, J.S.: On the efficiency of pseudo-marginal random walk Metropolis algorithms. Ann. Stat. **43**(1), 238–275 (2015)

Soto, A.: Self adaptive particle filter. In: IJCAI, pp. 1398–1406 (2005)

Tran, M.-N., Scharth, M., Gunawan, D., Kohn, R., Brown, S.D., Hawkins, G.E.: Robustly estimating the marginal likelihood for cognitive models via importance sampling. Behav. Res. Methods **53**(3), 1148–1165 (2020)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.