Ecological Modelling 455 (2021) 109608

Contents lists available at ScienceDirect

Ecological Modelling

journal homepage: www.elsevier.com/locate/ecolmodel

# Sequential Monte-Carlo algorithms for Bayesian model calibration – A review and method comparison☆

Matthias Speich [a,3], Carsten F. Dormann [a,2], Florian Hartig [a,b,*,1]

[a] Biometry and Environmental System Analysis, University of Freiburg, Freiburg, Germany
[b] Theoretical Ecology, University of Regensburg, Regensburg, Germany

ARTICLE INFO

*Keywords:*
Bayesian inference
Sequential Monte-Carlo (SMC)
Mechanistic models
Model calibration
Particle filters
Equifinality

ABSTRACT

Bayesian inference has become an important framework for calibrating complex ecological and environmental models. Markov-Chain Monte Carlo (MCMC) algorithms are the methodological backbone of this framework, but they are not easily parallelizable and can thus not make optimal use of modern computer architectures. A possible solution is the use of Sequential Monte Carlo (SMC) algorithms. Currently, SMCs are used mainly for Bayesian state updating, for example in weather forecasting, and are thought to be less efficient for parameter calibration than MCMCs. Unlike MCMCs, however, SMCs are easily parallelizable. Thus, SMCs may become an interesting alternative when modelers have access to parallel computing environments. The purpose of this paper is to provide an introduction to SMC algorithms for Bayesian model calibration, and to explore the trade-off between efficiency and parallelizability for MCMC and SMC algorithms. To that end, we discuss different SMC variants, and benchmark them against a state-of-the-art MCMC algorithm by calibrating three ecological models of increasing complexity. Our results show that, with appropriately chosen settings, SMCs can be faster than state-of-the-art MCMC algorithms when a sufficiently large number of parallel cores are available and when the model runtime is large compared to communication overhead for parallelization (on our hardware, a model runtime of 20 ms was enough to favor SMC algorithms). Efficient SMC settings were characterized by a balanced mix of SMC filtering and MCMC mutation steps, suggesting that mixing MCMC and SMC principles may be ideal for creating efficient and parallelizable calibration algorithms. The algorithms used in this study are provided within the BayesianTools R package for Bayesian inference with complex ecological models.

## 1. Introduction

Many research communities within ecology and environmental sciences traditionally rely strongly on complex, process-based system models (e.g. global circulation models / GCMs, (global) dynamic vegetation models (G/DVMs), or hydrological models, see Jeffers, 1982). Other ecological fields have recently moved in the direction of mechanistic modeling approaches, for models of biodiversity change (e.g. Urban et al., 2016), or models of biogeography or macroevolutionary processes (Pontarp et al., 2019). Through this increasing use complex process-based models, in particular for forecasting (e.g. Dietze 2017), the question of how to choose model parameters and determine their uncertainties has become more prominent.

Historically, parameters for complex process-based models were often chosen ad-hoc, either by direct measurements, or by literature reviews, without formal model calibrations (cf. Hartig et al., 2012; Dietze 2017). With sufficient computing power, however, process-based models can be fit in very much the same way as statistical models, and standard statistical procedures such as residual checks, model comparison or model averaging can be performed on top of the fitted model (e.g. van Oijen et al., 2005; Schoups and Vrugt, 2010).

A practical difference to statistical models, however, is that process-based models tend to be more nonlinear and connected, which, together with data-limitations, results in frequent occurrences of multiple optima

☆**Tweetable abstract:** We show that SMC algorithms can outperform MCMC algorithms for fitting slow models on parallel hardware. R package provided.

* Corresponding author.
*E-mail address:* florian.hartig@ur.de (F. Hartig).
[1] ORCID FH: 0000–0002–6255–9059
[2] ORCID CFG: 0000–0002–9835–1794
[3] ORCID MS: 0000–0003–0173–5351

https://doi.org/10.1016/j.ecolmodel.2021.109608
Received 21 October 2020; Received in revised form 21 April 2021; Accepted 11 May 2021
Available online 5 June 2021

or trade-offs between parameters (non-identifiability or equifinality). Equifinality means that there is no unique optimal parameter set, but that many combinations of parameters fit the observations equally well (see Beven and Freer, 2001). Moreover, unlike for statistical models, parameters in mechanistic models have a clear meaning and researchers often have prior expectations about their likely value. Optimization approaches often perform poorly for such nonlinear equifinal problems, and frequentist statistical approaches such as maximum likelihood estimation struggle to provide a framework for considering prior information about ecologically plausible parameter values in the calibration procedure. For all these reasons, Bayesian approaches have risen in popularity for calibrating complex process-based models in recent years, for example to model vegetation dynamics (Van Oijen et al., 2005; Lagarrigues et al., 2015), hydrological processes (Beven and Freer, 2001; Jeremiah et al., 2012, 2011; Zhu et al., 2018) or biogeochemical processes (Arhonditsis et al., 2008; Ahrens et al., 2014).

The aim of the Bayesian calibration procedure is to calculate the posterior uncertainty $p(\theta|y)$ for the model parameters. The posterior is interpreted as the probability density for a given parameter set $\theta$ to be correct, conditional on the observed data $y$ and our prior beliefs $p(\theta)$, the latter representing our knowledge about the likely parameter values beyond the calibration data. The ability to include prior information in a Bayesian calibration provides a framework to build upon pre-existing knowledge on the values of parameters. Conversely, where there is little empirical basis for constraining the values of a certain parameter, this can be accounted for by setting a wide prior distribution. The posterior is proportional to the product of the prior distribution $p(\theta)$, representing our best knowledge about parameter values before model calibration, and the likelihood function $p(y|\theta)$, the probability of observing the data $y$ given the model with parameters $\theta$. Informally, one can view the likelihood as the goodness-of-fit of the model with parameters $\theta$, and the Bayesian posterior as a mix between prior information and the information provided by the data through the likelihood. Because the Bayesian framework does not search for an optimal parameter combination, but only updates the prior uncertainty by the new data, there is no lower limit of how many data are needed for calculating a posterior estimate, nor pose multiple optima or equifinality a fundamental problem for a valid posterior estimate. For a more thorough description of the typical Bayesian calibration process, we refer to Van Oijen et al. (2005) or Hartig et al. (2012).

A practical challenge for using Bayesian inference is that the posterior can usually not be calculated analytically, and must therefore be approximated numerically, which is often computationally demanding. The current standard solution to this problem are Markov Chain Monte-Carlo (MCMC) methods (see Andrieu et al., 2003 for an overview). Very briefly, the idea of an MCMC is to specify a stochastic (Markov) process that performs a random walk in parameter space. The Markov process is chosen such that the probability of visiting each parameter combination is proportional to its posterior density $p(\theta|y)$. The simplest algorithm of this kind is the so-called Metropolis-Hastings MCMC. Many MCMC variants have been proposed to increase the efficiency this algorithm, for example through dynamically adapting the transition process (Haario et al., 2006), or combining principles of MCMC with evolutionary algorithms (ter Braak and Vrugt, 2008). Those variants have improved sampling efficiency (i.e. the number of iterations needed to get a good estimate of the posterior), but they all still require a substantial number of model evaluations to fit a complex model (often in the range of $10^6$ - $10^8$), which can make calibrations of complex models a matter of weeks or even months. More recent MCMC algorithms run several Markov chains concurrently (e.g. ter Braak and Vrugt, 2008; Vrugt et al., 2009), but because all MCMCs principally evaluate a new parameter proposal relative a current parameter set, calculations can be parallelized only to a very limited degree. This lack of MCMC parallelizability currently limits the use of Bayesian inference for calibrating models with a runtime of minutes or longer, and poses a serious problem for the wider adoption of Bayesian inference in the ecological modeling community.

Outside the context of model calibration, a widely-used alternative to MCMC sampling are Sequential Monte Carlo (SMC) algorithms (see Del Moral et al. (2006) for an extensive treatment of the mathematical aspects of SMC; and Speekenbrink (2016) for an introduction). The idea of an SMC is to generate a (usually large, e.g. $10^3$) population of parameter sets termed "particles", which are then iteratively filtered according to their fit to the data, in a way that the final particle population approximates $p(\theta|y)$. The main use of SMC methods are situations where it is advantageous to use data iteratively, for example in state-space models, or when data becomes available in real time, as in weather forecasting (Doucet and Johansen, 2009; Speekenbrink, 2016). However, SMCs can also be used to estimate parameters for statistical (Fan et al., 2008; Dufays, 2016) and dynamic models (Jeremiah et al., 2012, 2011; Sisson et al., 2007; Zhu et al., 2018). The attractive property of doing so is that, unlike MCMCs, this process can be parallelized, making it potentially suitable for calibrating complex, time-consuming models.

The downside of replacing MCMCs with SMCs is that the latter tend to be less efficient for a single model calibration than an MCMC and require careful tuning to avoid sample degeneracy and impoverishment (detailed in the methods below). Moreover, the Markovian nature of MCMCs integrates information from past model evaluations faster into the algorithm's exploration of the parameter space. Although many methods exist to reduce the risk of these problems (Jasra et al., 2011; Jeremiah et al., 2012; Li et al., 2014), those are not guaranteed to prevent their occurrence, and SMCs have therefore rarely been used for model calibration.

Whether this neglect of SMCs is justified, however, is an open question, in particular when considering that modern computer hardware is moving towards parallel architectures that cannot be efficiently used by MCMCs. So far, there have been few serious attempts to benchmark the two approaches against each other. An exception is Jeremiah et al. (2011), who found in a direct comparison between an SMC and an MCMC sampler that SMC was less likely to be trapped in local optima, leading to greater sample robustness. However, the Adaptive Metropolis MCMC (Haario et al., 2001) used in that study is known for its inferior performance in the presence of multiple optima, compared to more advanced MCMC algorithms (e.g. Haario et al., 2006; ter Braak and Vrugt, 2008). Overall, the relative performance of SMC and MCMC samplers is therefore still somewhat unresolved. A key question, so far not systematically investigated, is whether the parallelizability of SMC samplers could offset possible disadvantages regarding efficiency in practical situations.

Here, we address these issues by first giving an overview of the design principles and flavors of SMC algorithms, identifying the most promising SMC variants for parameter calibration. We then implement an SMC algorithm with the possibility to modify various SMC tuning parameters and variants and evaluate their ability to calibrate three models of increasing complexity against a state-of-the-art MCMC sampler as a benchmark. The considered models were a multivariate normal distribution, a simple dynamic vegetation model, and a state-of-the-art forest growth model. We assessed the approximation error of the multivariate posterior distribution against the time used by the algorithm.

## 2. Methods

### 2.1. A primer to SMC sampling

Sequential Monte Carlo (SMC) is a generic term for a range of algorithms based on similar principles. SMCs, also known as particle filters, are always initialized by generating a population of $N$ particles (each particle corresponding to a possible parameter set), typically drawn from some initial distribution. The SMC will now *filter* these particles in a way that the final distribution of particles approximates the posterior distribution. This filtering typically consists of three steps: weighting, resampling and mutation (illustrated on Fig. 1a), which are

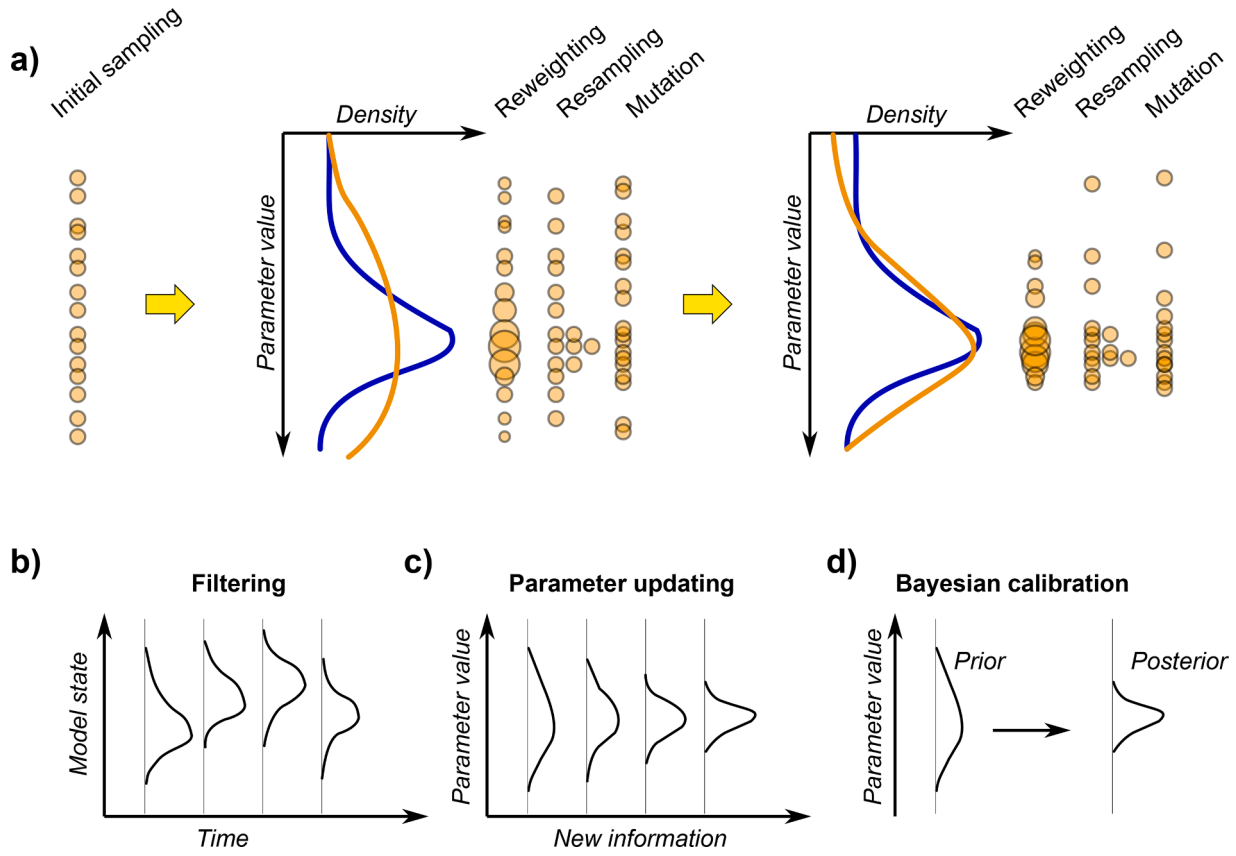


**Fig. 1.** Principle of an SMC algorithm (a) and its different use cases (b-c). Panel (a) illustrates that, as for an MCMC, the objective of an SMC is to efficiently sample from a (typically high-dimensional) target distribution (blue line). The SMC achieves this by starting with a sample from an initial distribution, and then to perform resampling and mutation steps on this sample based on a number of intermediary distributions (orange lines), so that the final sample converges to the posterior. The purpose of resampling is to change the distribution toward the target, while the purpose of mutating is to maintain sufficient diversity in the sample (details see text). Panels (b-d) show three use cases for SMC algorithms in Bayesian computations: updating the uncertainty of state variables as temporal information enters the model over time, for example in weather forecasting (b), sequential inclusion of new information in a parameterization (c) and a standard Bayesian calibration, where all data is used at once (d). Note that the most typical use for an MCMC is case (d), while the most typical use for an SMC is case (b).

iteratively applied while data is added at each iteration.

Weighting means that each particle is assigned a weight, usually proportional to its likelihood (i.e. fit to the data). In the next step, the particles are resampled with replacement, with their sampling probability given by their normalized weights. This means that particles with low weights (= bad fit) tend to be discarded, whereas particles with high weights (= good fit) are more likely to be replicated. The purpose of this step is preventing the issue of particle or sample degeneracy (Li et al., 2014), i.e. the concentration of weights on a small number of particles. Finally, to avoid having identical particles, a mutation step is performed where particles are slightly moved in parameter space based on a defined transition rule. This alleviates the risk of "sample impoverishment" (Li et al., 2014), i.e. the situation where nearly all particles have identical parameter values.

The typical use case for SMC algorithms are iterative filtering problems (Fig. 1b), for example for estimating time-varying model states, such as the location of a tracked object (Djuric et al., 2003) or soil moisture (Moradkhani et al., 2005; Vrugt et al., 2013). They may also be used to update model parameter values if new observations become available in real time (Fig 1c, for an example see e.g. Speekenbrink, 2016). However, SMCs can also be used to fit static model parameters (Fig. 1d), which is our interest in this study.

### 2.2. Using SMC algorithms for model calibration

As explained above, the principle of an SMC is to iteratively filter a population of particles regarding their fit to new information. Unlike in traditional SMC applications (Fig. 1b,c), data in a model calibration do not have a natural time order or appear iteratively. One could still establish an iterative filtering by adding the data step-wise to the SMC, but a more flexible approach is to use all data at once, and instead increase the weight of the data step-wise during the SMC iterations. Technically, this is done by creating a sequence of intermediary distributions moving gradually from an initial distribution $\pi_0$ to the posterior distribution, in a so-called bridge approach (Neal 2001; Fan et al., 2008). In the few instances where SMC was applied to model calibration (e.g. Fan et al., 2008; Jeremiah et al., 2012, 2011; Zhu et al., 2018), this approach was always used.

In practice, the initial distribution $\pi_0$ is usually the prior, although any distribution can be chosen in principle. The intermediate distributions $\pi_n$ are then defined as:

$$\pi_n \propto \pi_0(\theta)^{1-\beta_n} \pi(\theta|y)^{\beta_n}, \quad (1)$$

where the sequence of parameters $\beta_n$, defined so that $0 \leq \beta_1 \leq \beta_2 \leq \ldots \leq \beta_n = 1$, controls the mix between the initial distribution and the final posterior based on all data. When choosing a sequence $\beta_n$, a trade-off exists between stability (avoiding particle degeneracy) and efficiency (speed). If the difference between the intermediate distributions is too great, filtering will be too aggressive, leading to particle degeneracy (Jeremiah et al., 2011). If the difference is too narrow, the algorithm will perform more iterations than necessary. To solve this trade-off, Jeremiah et al. (2012) proposed an adaptive approach, starting with a pre-defined series $\beta_n$, and decreasing the spacing if the algorithm is at

risk of becoming unstable.

Mutating the particles after resampling is crucial for the efficiency of SMC samplers (Zhu et al., 2018). For SMCs designed for parameter calibration, some variation of the Metropolis algorithm (Metropolis et al., 1953) is typically used, which also forms the basis of many MCMC samplers. The principle of the Metropolis move is that a new parameter value is generated based on the current one according to some proposal function. The probability to accept the move is given by the "Metropolis-ratio": the ratio of $p(\theta|y)$ for the proposed over the current parameters. In an SMC mutation step, this procedure is applied to each particle, possibly multiple times per iteration. Indeed, the number of MCMC mutation steps per SMC iteration has been shown to have a substantial effect on sampler efficiency (Fearnhead and Taylor, 2013).

The simplest Metropolis MCMCs use pre-defined univariate normal distributions as proposal distributions. Many of the improvements over this simple choice in the context of MCMC (e.g. Haario et al., 2006; ter Braak and Vrugt, 2008) have also been applied within SMC samplers. For example, the covariance structure of a multivariate proposal distribution can be updated at each iteration, based on the covariance matrix of the particles (Chopin, 2002; Fan et al., 2008; Jeremiah et al., 2012). Another example is Zhu et al. (2018), who use a combination of genetic and evolutionary algorithms to generate proposals, leading to a considerable increase in sampler efficiency.

### 2.3. Description of the smc algorithm used in this study

The algorithm proposed in this study (Fig. 2a) was based on the lessons drawn from our literature review regarding efficient SMC implementations for model calibration. It uses an adaptive scheme to determine the series of intermediate distributions (following Jasra et al., 2011). The sampler is initialized by drawing $N$ particles from an initial distribution $\pi_0$. The initial exponent $\beta$ (which controls the position between the initial and the final distribution) is set to zero. As the initial population represents an unweighted sample from $p_0$, all weights are set to $1/N$. At each iteration, the algorithm calculates the particle weights and then searches for a new value $\beta'$ based on an informal measure of particle diversity, the effective sample size $N_{\text{eff}}$

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} w_i^2}, \tag{2}$$

where $w_i$ are the normalized weights of the $N$ particles and takes values between 1 (when all the weight is concentrated on a single particle) and $N$ (when all particles have equal weight). The algorithm will determine the value of $\beta'$ for which the effective sample size $N'_{\text{eff}}$ equals a fraction $a$ of the current $N_{\text{eff}}$. One can think of this choice as a trade-off between stability (large effective sample size) and efficiency (large $\beta'$ steps to minimize the number of filtering steps).

Next, a resample-move step is made: the particles are first resampled using systematic resampling (see Douc and Cappe, 2005) and all weights

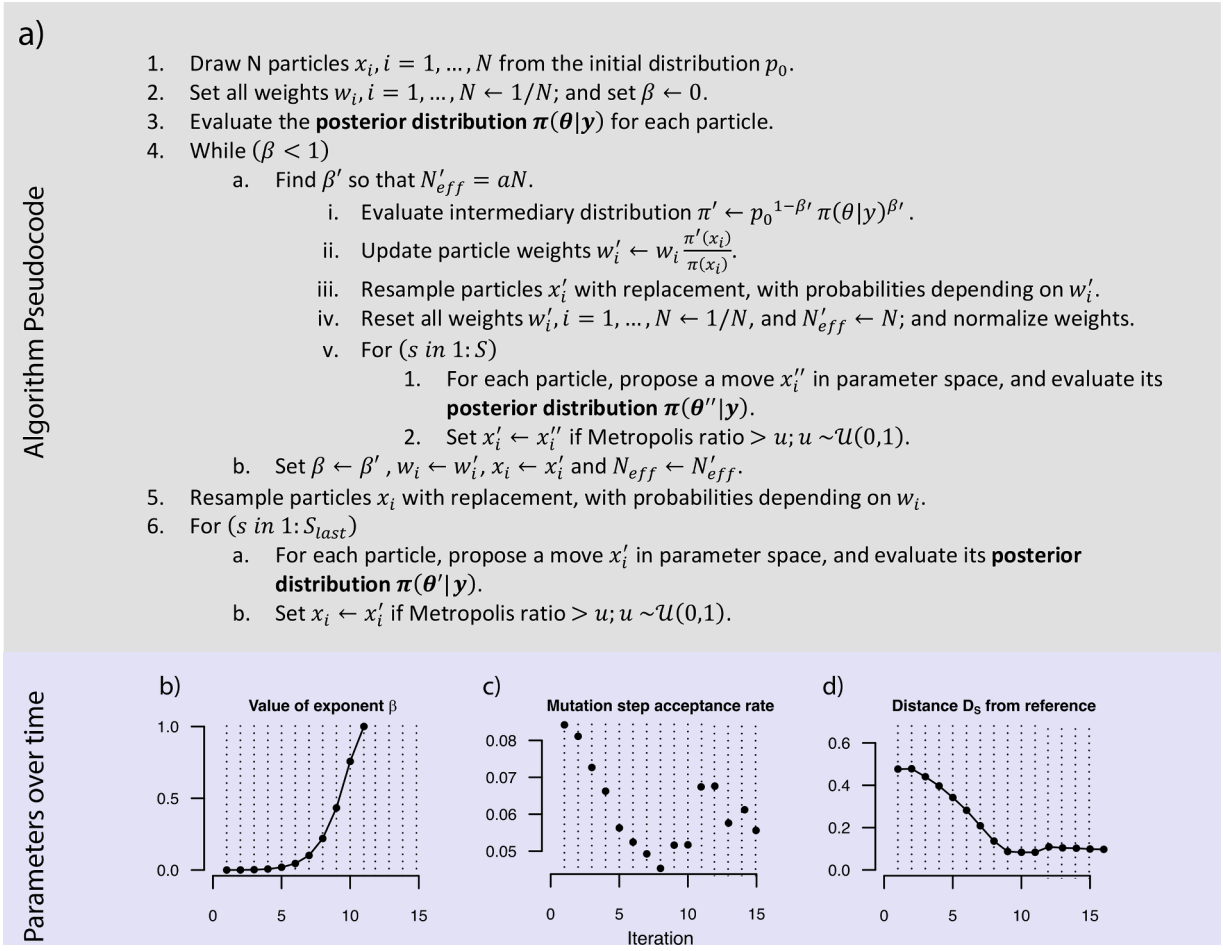


**Fig. 2.** a) Pseudocode for the SMC sampler proposed in this paper. The parts in bold highlights at which point the entire model has to be evaluated b-d) change of parameters and acceptance rate during the runtime (iterations) of the SMC for an example simulation. b) change of the exponent β, which controls the gradual change from the prior to the posterior weighting of the particles during the SMC steps. c) acceptance rate of the mutation step d) distance $D_s$ between the true and the approximate posterior distribution. The exact definition of the distance measure is described in section "Benchmarking the SMC against a state-of-the-art MCMC".

are reset to $1/N$. Next, the particles are mutated by an MCMC step based on Differential Evolution and snooker update (ter Braak, 2006; ter Braak and Vrugt, 2008): for each particle $x_i$, two particles $x_j$ and $x_k$ are randomly sampled, with $i \neq j \neq k$. The proposal $x_i^{'}$ is obtained as follows:

$$x_i^{'} = x_i + \gamma(x_j - x_k) + \varepsilon, \tag{3}$$

where $\gamma$ is a scaling factor, typically set to $2.38/\sqrt{2d}$ (ter Braak, 2006), with $d$ being the number of parameters under calibration, and $\varepsilon$ is drawn from a narrow distribution $\mathcal{N}(0,\ b)$, with a default of $b = 10^{-4}$. In 10% of the cases (randomly selected), a snooker update is performed instead, following ter Braak and Vrugt (2008). As $b$ implies a particular scale for the parameters, $b$ should be either adjusted to the prior or expected posterior parameter uncertainty, or parameters should be scaled accordingly.

When $\beta$ reaches 1 (meaning that the full data is used), a last resampling step is performed (Step 5), so that the particles can be considered an unweighted sample from $\pi(\theta|y)$. As this sample likely contains many identical particles, a final round of mutation steps (Step 6) are executed to increase sample diversity.

During the search for $\beta^{'}$ (Step 4a), some weights may take infinite values, due to $p^{'}(x_i)/(x_i)$ ratios exceeding the numeric accuracy of the computer. In this case (omitted from the listing in Fig. 2), a resample-move step is executed, and the current value for $\beta$ is kept for the next iteration. This is a failsafe to keep the sampler from crashing due to numerical issues.

The output of an exemplary SMC simulation is displayed on Fig. 2b. The emerging series of $\beta$ values reveals that small changes between intermediary distributions are necessary at the beginning to avoid sample impoverishment, and while $\beta$ increases these changes become larger as the SMC progresses. The acceptance rate of the MCMC steps (averaged over all $S$ MCMC steps at each iteration) fluctuates between 6 and 9% in this example. The distance $D_S$, a measure of dissimilarity to the true posterior, is defined below.

### 2.4. Benchmarking the SMC against a state-of-the-art MCMC

We benchmarked the performance of the SMC algorithm in four calibration case studies of increasing complexity against a Differential Evolution MCMC with snooker update ($DE_{ZS}$; ter Braak and Vrugt, 2008), which is a state-of-the-art MCMC algorithm. As performance measure, we calculated the similarity between SMC or MCMC samples to a reference distribution using the following metric $D_S$, proposed by Laloy and Vrugt (2012):

$$D_S = \sqrt{\frac{1}{2d}\sum_{i=1}^{d}\left[\left(\frac{\mu_{i,1}-\mu_{i,2}}{\sigma_{i,1}}\right)^2 + \left(\frac{\sigma_{i,1}-\sigma_{i,2}}{\sigma i_{,1}}\right)^2\right]}, \tag{4}$$

where $\mu_{i,x}$ and $\sigma_{i,x}$ ($x \in \{1,2\}$) are the mean and standard deviation of the marginal distribution of each parameter in the reference sample ($x = 1$) and the sample being evaluated ($x = 2$), and $d$ is the number of parameters. Essentially, $D_S$ expresses the average normalized Euclidean distance between the means and standard deviations of both distributions. As reference distribution, we used the analytical posterior when it was available, and otherwise an extremely long independent DEzs run.

As case studies for the benchmark, we selected a multivariate normal distribution, a simple ecosystem model, and a state-of-the-art forest growth model. As shown in Table 1, these case studies differ greatly in model complexity. The first example is a simple three-dimensional multi-variate normal distribution, with a strong correlation ($r \cong 0.9$). The second case study consisted of calibrating the Very Simple Ecosystem Model (VSEM), a simple ecosystem model serving as benchmark in R's BayesianTools package. Two configurations were tested with VSEM: in one case, we included two parameters in the calibration for which the posterior was strongly correlated ($r \cong 0.95$), whereas in the second case, there were no strong posterior correlations between parameters. The third case study used the model 3-PGN (Minunno et al., 2018) , a recent extension of the forest growth model 3-PG (Landsberg and Waring, 1997). With 51 parameters, 3-PGN is substantially more complex than the VSEM model, but due to its monthly time step, the model still executes very fast (Table 1). To emulate the calibration of a computationally more expensive model, a fourth case study ($3\text{-PGN}_{sleep}$) was set up, in which the execution time of 3-PGN was increased by adding a pause of 50 ms after each model execution. All other settings are identical.

In case of the multivariate distribution, the distribution itself was the calibration target. For the dynamic models, we simulated synthetic data from the model's default parameters and added a normally distributed observation error on top, and specified a likelihood from these assumptions, including parameters for the error terms (1 for VSEM and 2 for 3-PGN, because two outputs were used for calibration in the latter case). For all models, we used wide uniform (uninformative) priors. As min / max values for the priors, we chose [−5,5] for the multivariate normal distribution, and for the VSEM / 3-PGN model, we chose the min / max values for calibration provided by the model developers (see supplementary material for details).

To explore how the SMC's control parameters influence the sampler's efficiency, we ran all case studies in a full factorial design, varying the values of the control parameters (Table 2), resulting in 240 alternative SMC configurations. An exception is the multivariate normal case, where $N$ was set to 50, 100 and 1000 particles, yielding 180 algorithm configurations that we tested. For each model, the experiment

**Table 2**
Settings used in the evaluation of the SMC algorithm.

| Control parameter | Description | Values |
|---|---|---|
| $N$ | Number of particles | 5000, 20 000, 50 000, 100 000 |
| a | $N_{eff}$ threshold for resample-move step (fraction of $N$) | 0.5, 0.75, 0.9 |
| $S$ | Number of mutation steps | 2, 5, 10, 20, 30 |
| $\gamma$ | Scaling factor for mutation steps | 0.01, 0.1, 0.333, 0.5 |

**Table 1**
Overview of the five case studies described in this paper. Execution time indicates the average run time of a single model run on the cluster where these experiments were performed. As explained in the main text, the $DE_{zs}$ can be partly parallelized, albeit to a much lower degree than an SMC algorithm. We tested for all case studies if such a parallelization speeds up computations and always selected the faster option.

| Case study | Number of calibrated model parameters (+ error terms) | Average runtime for a single model run [ms] (SD in brackets) | Iterations for reference MCMC-$DE_{zs}$ | Parallel MCMC-$DE_{zs}$ |
|---|---|---|---|---|
| Multivariate normal | 3 | 0.56 (0.49) | 100 000 | No |
| VSEM (uncorrelated) | 6 + 1 | 0.37 (0.48) | 2 000 000 | No |
| VSEM (correlated) | 6 + 1 | 0.37 (0.48) | 2 000 000 | No |
| 3-PGN | 51 + 2 | 0.93 (0.29) | 2 000 000 | No |
| $3\text{-PGN}_{sleep}$ | 51 + 2 | 51.39 (0.5) | 2 000 000 | Yes |

was repeated five times with different random seeds, to assess convergence of the algorithm and variations in runtime and sample quality. Convergence was assessed using the Gelman-Rubin diagnostic $\widehat{R}$ (Brooks and Gelman, 1998; Gelman and Rubin, 1992). It was assumed that convergence is reached when no univariate $\widehat{R}$ for point estimates and upper confidence interval is greater than 1.05, and the multivariate $\widehat{R}$ estimate (Brooks and Gelman, 1998) is less than 1.2.

The MCMC-DE$_{ZS}$ algorithm used as a benchmark requires several chains to be run independently from each other. Based on the results of ter Braak and Vrugt (2008), the number of chains was set to 3. Convergence was assessed by estimating $\widehat{R}$ over the 3 chains of a run, with the same criteria as for the SMC runs. The number of iterations, given in Table 1, was selected to ensure convergence of the reference run. Because each single MCMC iteration depends on the previous iteration, MCMCs iterations can in general not be computed in parallel. However, for the MCMC-DE$_{ZS}$, each MCMC iteration requires several model evaluations to calculate the posterior values for the (in our case) three internal DE chains. These internal computations will make the algorithm somewhat faster (at maximum by a factor 3), unless the model evaluates very fast. In this case, parallelization overhead will offset parallelization gains. For all case studies, we tried both options (parallel and sequential), selected the faster option for the benchmark (Table 1). In general, we note that although the MCMC-DE$_{ZS}$ offers some options for parallelization, these are very limited, because only a small number of computer cores can sensibly be used in parallel with this algorithm. SMCs, however, scale practically linearly with the number of cores, up to a large number of cores (see our later results, Fig. 4), which is one of their core advantages over MCMCs.

### 2.5. Effect of parallelization

All the experiments described above were executed on the same computer cluster (NEMO at the University of Freiburg; technical specifications see supplementary material). Each node on this cluster consists of 20 cores. For the SMC experiments, 40 cores were used, spanning no more than 2 different nodes. To assess the effect of parallelization on runtime, further tests were run on a varying number of cores (1, 2, 5, 10, 20 and 40 cores). In these tests, the SMC parameters were set to $N = 50\,000$, $a = 0.9$, $S = 30$ and $\gamma = 0.1$. For each of these settings, 3-PGN was

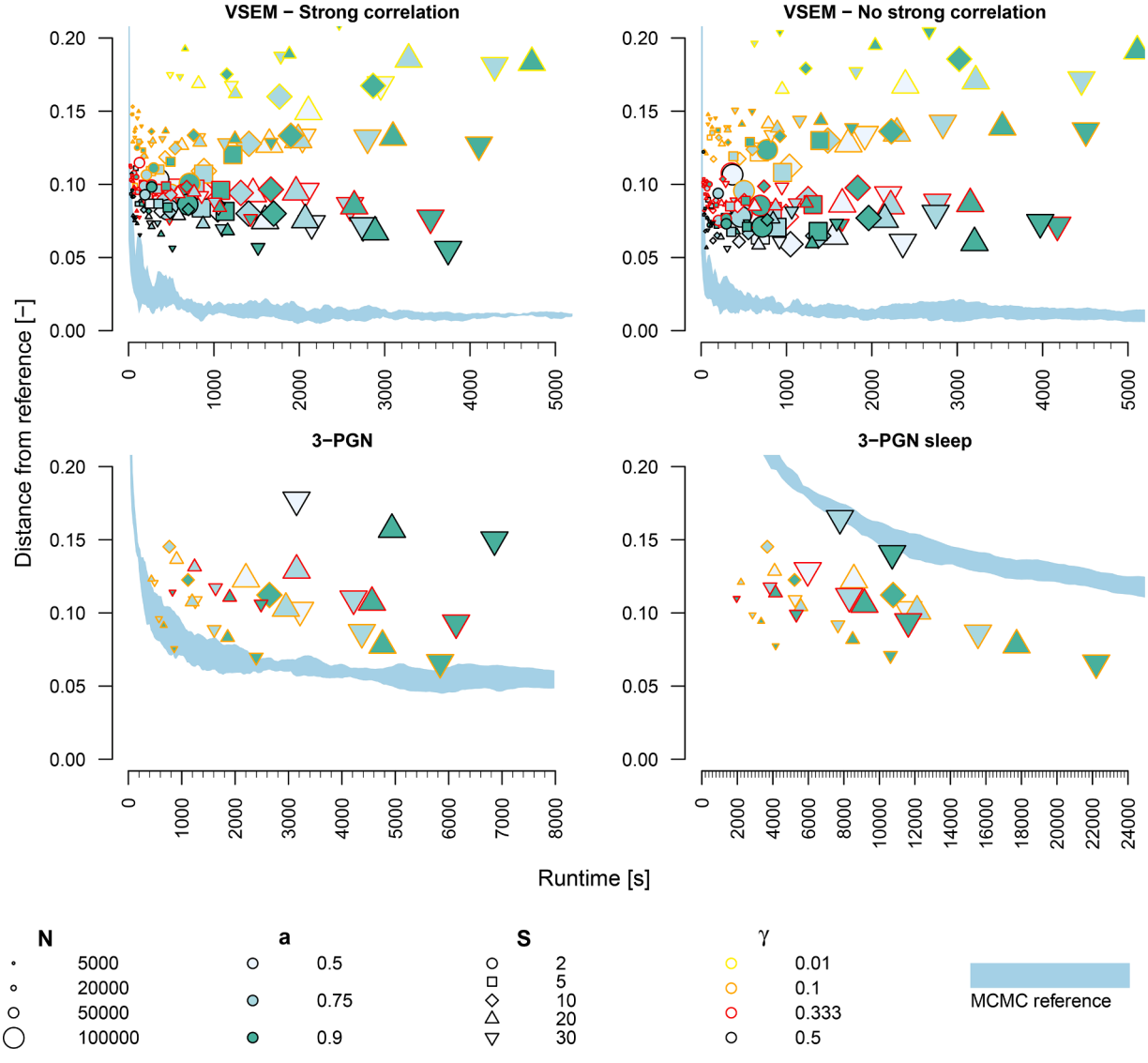


**Fig. 3.** Comparison of MCMC and SMC sampler quality for the dynamic models VSEM and 3-PGN, and different settings for the SMC sampler. The shaded area shows the difference to the reference $D_S$ for five MCMC runs as a function of the runtime. The points show the average runtime and $D_S$ for five SMC runs with equal algorithm settings coded by size (number of particles N), fill color (intermediate distribution tuning parameter a), symbol (final mutation steps S) and border color (proposal scaling parameter), respectively.

run without artificial runtime increase, as well as with a runtime increase of 20 and 50 ms. Each combination was executed five times, to account for potential variations in execution time.

## 3. Results

### 3.1. Quality and efficiency

For the multivariate normal case (Fig S1 in the supplementary material), MCMC runs converged after 22.8 s on average and achieved a $D_S$ score of 0.0018. 83 out of 180 SMC runs reached convergence, and the lowest $D_S$ score is 0.028. Execution time for the converging SMC runs ranged between 0.35 and 36.32 s. SMC runs were more likely to reach convergence with a greater number of particles and mutation steps.

The results for the more complex models are summarized in Fig. 3. For the MCMC, the distance from the reference run was monitored every 1000 iterations of the algorithm. The blue area indicates the spread of the five MCMC runs. For SMC, only converged runs are shown on the graph. SMC runs located below the shaded area compare favorably to the MCMC algorithm, i.e. the same sample quality can be achieved in less time.

In both VSEM examples, the MCMC sampler quickly reached a state close to the reference ($D_S < 0.01$). Most of the SMC runs converged according to the Gelman-Rubin criterion (180 and 166 out of 240 for the cases with and without strong correlation, respectively), although the sample quality, as expressed by $D_S$, varied greatly. None of the SMC runs are located below the MCMC curves, meaning that they did not outperform the MCMC. The plot shows a strong dependency of sample quality on the scaling factor of the MCMC proposals $\gamma$: the higher the value for $\gamma$, the better the sample quality. Interestingly, more particles by themselves did not translate into better performance.

For the 3-PGN benchmark, the MCMC sampler reached levels of $D_S$ as low as 0.04. Of the 240 SMC runs, 32 reached convergence. All those that did had a at least 10 mutation steps $S$, and for most, the value of $\gamma$ was either 0.1 or 0.333. Some of the SMC runs were located at the same level as the MCMC curves, indicating a similar efficiency. Sample quality generally increased ($D_S$ decreased) with increasing $S$, at the expense of additional runtime. Increasing the number of particles $N$ generally had a positive effect on the performance of the algorithm, but this effect was often small. Runtime, in contrast, increases greatly with $N$. For the three runs with $a = 0.9$, $\gamma = 0.1$ and $S = 30$, for example, $D_S$ only improved marginally with $N$, whereas runtime was six times higher for $N = 100\,000$ than for $N = 20\,000$.

The ranking of the different SMC settings stayed qualitatively similar for the benchmarks with 3-PGN$_{sleep}$, compared to 3-PGN, which is unsurprising, given that the two models are functionally identical and differ only in their runtime. In this case, 29 of the SMC settings achieved convergence. Small differences between the ranking of different settings for 3-PGN$_{sleep}$ and 3-PGN are likely due to the inherent stochasticity of the sampler. For the 3-PGN$_{sleep}$ model, all converged SMC runs were at the same level or below the MCMC curves, indicating that the SMC sampler was typically more efficient than the MCMC sampler for this model.

### 3.2. Effect of parallelization

To better understand the trade-off between communication overhead (i.e. the computation time lost during parallelization due to the necessary communication between cores) and speed-up due to parallelization, we performed SMC simulations with models of different runtime across a range of used CPU cores, and regressed the relationship between runtime and cores with a linear regression on the log-log scale (Fig. 4).

For the standard 3-PGN model (runtime approx. 1 ms), parallelization could reduce the total SMC runtime maximally by a factor 3 when using 10 cores instead of running the algorithm serially (Fig. 4, left). Runtime increased with less than linearly with the number of cores (log-log slope estimate −0.5), and adding more cores beyond 10 did not increase a further speed-up. Scaling was much better for the calibrations of the identical 3-PGN model which was modified to include a pause of 20 and 50 ms after model execution (Fig. 4, middle, right). The log-log slope estimates were −0.88 and −0.94, suggesting that the scaling would likely approach the ideal value of −1 (i.e. a runtime ∼ 1/cores scaling) for even slower models. A detailed regression table with slope estimates is provided in the supplementary material, Fig. S2.

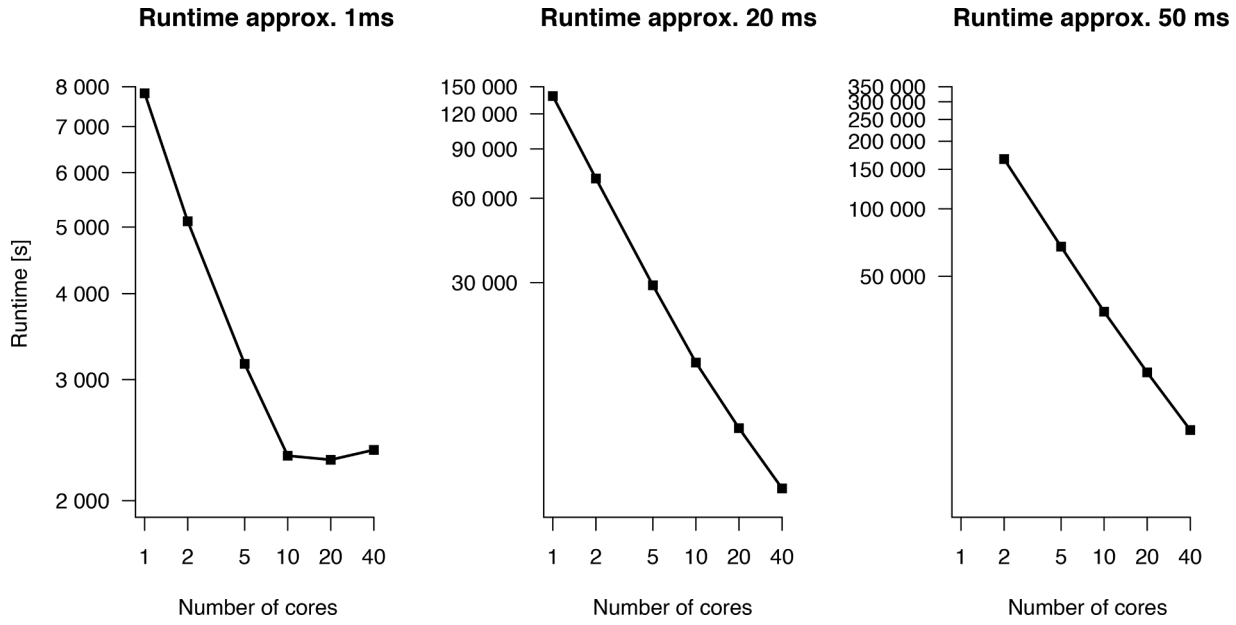


**Fig. 4.** Scaling of the SMC runtime with the number of cores used for the standard 3-PGN model (left, runtime approx. 1 ms) and the identical model with added execution pauses of 20 and 50 ms (middle, right). Each point displays the average runtime of 5 SMC calibrations. The straight solid (black) line shows the fit of a linear regression to the data points (for the left figure, the points with 20 / 40 cores were omitted). Dashed (red) show slopes of −1, which, in a log-log plot, display an ideal linear scaling of the runtime with the number of cores.

## 4. Discussion and conclusions

The purpose of this paper was to provide an introduction into SMC sampling for the calibration of complex environmental models, to explore which SMC settings work well for complex dynamic model, and to assess the potential runtime reduction through parallelization for SMC algorithms compared to MCMC methods.

Our main findings are that SMC samplers are efficient algorithms for model calibration that can outperform MCMCs for runtime-intensive models, when at least a moderate number of parallel computer cores (in the order of some 10 cores) is available to the user (see Figs. 3,4). A possible drawback of SMCs is that convergence and sampling efficiency depended strongly on the tuning parameters of the algorithm. For the best settings of the tuning parameters, we could achieve relatively reliable convergence, but badly tuned algorithms often showed particle degeneracy and other problems. Once these phenomena occur, it is difficult for the algorithm to recover, unlike for an MCMC, which can always be run longer, even if it mixes badly. It is therefore vital that SMC algorithms are correctly tuned from the start. Assuming that the results from our test cases can be extrapolated to other ecological models, we would conclude that a large number of particles is less important than a sufficient number of intermediate MCMC steps to achieve favorable convergence behavior. The SMC settings that were most efficient in our simulations effectively prescribe an algorithm that consists of a mix between MCMC and SMC steps. We conjecture that such "mixed algorithms" may be an interesting direction for future research.

The values for the tuning parameter $\gamma$ (scaling factor for the DE MCMC steps during mutation) that led to greatest sampling efficiency were close to the optimal value for normal distributions in standard MCMC tuning ($\gamma = 2.38/d^{1/2} \rightarrow 0.9$ for VSEM and 0.3 for 3-PG), suggesting that many of the lessons learned from MCMC tuning can be taken over to the mutation steps in an SMC. Given the apparent importance of the mutation steps, it seems obvious that further research should be directed towards optimizing this part of the algorithm. A particularly promising idea seems to us to examine adaptive procedures to determine the number of MCMC steps, as suggested by Fearnhead and Taylor (2013).

Whether SMC calibration can outperform MCMC calibration depends primarily on the trade-off between communication overhead and parallelizability (Fig. 4). In the case of the standard 3-PGN model without artificial runtime increase, the fastest SMC configurations came close to, but did not beat, the time taken by the MCMC-DE$_{ZS}$ algorithm to reach the same sample quality, expressed as $D_S$ (1700s, cf. Fig. 3). By contrast, for the 3-PGN with a 50 ms pause, executing the SMC sampler with 5 or more cores took less time than using MCMC-DE$_{ZS}$ for the same sample quality. Communication overhead is also the reason why we used the possible parallelization option for the DEzs MCMC algorithm in the BayesianToools R package (which would have allowed using 3 CPU cores) only for the 3-PGN$_{sleep}$ scenario – for all faster models, running the MCMC without parallelization was faster, due to communication overhead. Given that we parallelized on a setup that has in our experience a comparatively low communication overhead, we conclude that a model runtime of around 20 ms – 50 ms is needed to make the use of an SMC worthwhile. When parallelizing across more cluster nodes, or in a distributed system, communication overhead will likely be larger, and models should therefore likely be even slower until the advantages of parallelization are offset by their costs. Given that many ecological and environmental models have runtimes of minutes or even hours, however, this does not appear to pose a major limitation for the applicability of SMC algorithms to model calibration.

Our results are less favorable for SMCs than those of Jeremiah et al. (2011), which we attribute to the fact that we benchmark against a state-of-the-art MCMC algorithm, rather than against a Metropolis-Hastings MCMC. Although we anticipate that our results are fairly representative, in that SMC will eventually outperform MCMC regarding given sufficient parallel resources, it would be interesting to run further tests, varying both the models and parameters under calibration, as well as the algorithm settings. In particular our results that favor a balanced mix of SMC and MCMC steps points towards exploring more aggressive design principles that mix SMC and MCMC ideas (e.g. Zhu et al., 2018)

Another issue that seems to require urgent attention are robust adaptation procedures that would, for example, automatically select the SMC settings that we identified as most efficient in Fig. 3, and reliable convergence checks that monitor and report potential issues during an SMC run. While there is some development in this direction (for example our automatic adaptation of the $\beta'$ steps), we find that SMCs lack in this respect compared to state-of-the-art MCMC algorithms such as the DEzs, which is completely self-adaptive under mildly beneficial circumstances.

Although we do provide the posterior parameter estimates in the code repository (see section Data accessibility), we stress that the purpose of this study was not to test the ecological plausibility of the fitted models or the identifiability of their parameters with real data (for an ecologically interpretable calibration of 3-PGN with real forest data, see, for example, Trotsiuk et al., 2020). Moreover, the use of synthetic data circumvented possible problems that can arise when calibrating models with structural error (e.g. Oberpriller et al., 2021). Such problems can potentially increase or decrease convergence speed of MCMCs or SMCs in practical applications. As we only compare across MCMC and SMCs with different settings in a fixed scenario, however, we do not see this as a potential limitation for our results.

## 5. Conclusions

In conclusion, our paper demonstrates that SMC algorithms should be considered as an alternative to MCMC samplers by modelers who want to calibrate slow models and have access to appropriate parallel computing hardware. For slow models, communication overhead due to parallel computing becomes negligible, and SMCs can draw on the opportunity provided by their superior parallelizability. Our results also suggest that SMCs with a large share of internal MCMC steps work best, possibly supporting a broader insight that a mix of both design principles (as it is already done to a lesser degree in population-based MCMC algorithms where several MCMC chains are run in parallel, such as the DEzs MCMC that we used as a reference) might actually be the most successful strategy for calibrating complex models on modern computer hardware. Our study also highlighted various opportunities for further research, in particular with regard to further development of robust adaptive methods for tuning SMC parameters, and the need to develop better tools for monitoring efficiency and convergence issues in SMCs.

## Data accessibility

The SMC algorithm presented in this paper is based on the BayesianTools R package, which is available on CRAN (Hartig et al., 2019). The BayesianTools package is a framework for Bayesian inference that includes a several MCMC samplers, as well as plots and diagnostic functions for Bayesian computations. The BayesianTools vignette (available via https://cran.r-project.org/web/packages/BayesianTools/vignettes/BayesianTools.html) explains in detail how to specify likelihoods and prior in the format that is expected by the package . Installation instructions and code for reproducing the results of this paper are available at https://github.com/florianhartig/BayesianTools/tree/master/Publications/SpeichEtAl-SMCForModelCalibration.

## CRediT authorship contribution statement

**Matthias Speich:** Methodology, Software, Formal analysis, Project administration, Writing – original draft. **Carsten F. Dormann:** Conceptualization, Writing - Review & Editing. **Florian Hartig:**

Conceptualization, Methodology, Project administration, Writing - Review & Editing.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

We gratefully acknowledge support by the DFG (DOR 786/8–1) as part of the Priority Program 1374 “Infrastructure-Biodiversity-Exploratories”. Johannes Oberpriller provided helpful comments on the presentation of the manuscript. We are grateful to the comments and suggestions by two anonymous reviewers.

## Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ecolmodel.2021.109608.

## References

Ahrens, B., Reichstein, M., Borken, W., Muhr, J., Trumbore, S.E., Wutzler, T., 2014. Bayesian calibration of a soil organic carbon model using Δ14C measurements of soil organic carbon and heterotrophic respiration as joint constraints. Biogeosciences 11, 2147–2168. https://doi.org/10.5194/bg-11-2147-2014.

Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I., 2003. An introduction to MCMC for machine learning. Mach. Learn. 50, 5–43. https://doi.org/10.1023/A:1020281327116.

Arhonditsis, G.B., Papantou, D., Zhang, W., Perhar, G., Massos, E., Shi, M., 2008. Bayesian calibration of mechanistic aquatic biogeochemical models and benefits for environmental management. J. Marine Syst. 73 (1–2), 8–30. https://doi.org/10.1016/j.jmarsys.2007.07.004.

Beven, K., Freer, J., 2001. Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the GLUE methodology. J. Hydrol. 249, 11–29. https://doi.org/10.1016/S0022-1694(01)00421-8.

Brooks, S.P., Gelman, A., 1998. General methods for monitoring convergence of iterative simulations. J. Comput. Graph. Stat. 7, 434–455. https://doi.org/10.1080/10618600.1998.10474787.

Chopin, N., 2002. A sequential particle filter method for static models. Biometrika 89, 539–552. https://doi.org/10.1093/biomet/89.3.539.

Del Moral, P., Doucet, A., Jasra, A., 2006. Sequential Monte Carlo samplers. J. R. Stat. Soc. Ser. B Stat. Methodol. 68, 411–436. https://doi.org/10.1111/j.1467-9868.2006.00553.x.

Dietze, M.C., 2017. Ecological Forecasting. Princeton University Press.

Djuric, P.M., Kotecha, J.H., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M.F., Miguez, J., 2003. Particle filtering. IEEE Signal Process. Mag. 20, 19–38. https://doi.org/10.1109/MSP.2003.1236770.

Douc, R., Cappe, O. (2005) Comparison of resampling schemes for particle filtering. IEEE, 64–69. 10.1109/ISPA.2005.195385.

Doucet, A., Johansen, A.M., 2009. A tutorial on particle filtering and smoothing: fifteen years later. In: Crisan, D., Rozovsky, B. (Eds.), Handbook of Nonlinear Filtering. Cambridge University Press, Cambridge.

Dufays, A., 2016. Evolutionary Sequential Monte Carlo samplers for change-point models. Econometrics 4, 12. https://doi.org/10.3390/econometrics4010012.

Fan, Y., Leslie, D.S., Wand, M.P., 2008. Generalised linear mixed model analysis via sequential Monte Carlo sampling. Electron. J. Stat. 2, 916–938. https://doi.org/10.1214/07-EJS158.

Fearnhead, P., Taylor, B.M., 2013. An adaptive Sequential Monte Carlo sampler. Bayesian Anal 8, 411–438. https://doi.org/10.1214/13-BA814.

Gelman, A., Rubin, D.B., 1992. Inference from iterative simulation using multiple sequences. Stat. Sci. 7, 457–472. https://doi.org/10.1214/ss/1177011136.

Haario, H., Laine, M., Mira, A., Saksman, E., 2006. DRAM: efficient adaptive MCMC. Stat. Comput. 16, 339–354. https://doi.org/10.1007/s11222-006-9438-0.

Haario, H., Saksman, E., Tamminen, J., 2001. An adaptive metropolis algorithm. Bernoulli 7, 223–242. https://doi.org/10.2307/3318737.

Hartig, F., Dyke, J., Hickler, T., Higgins, S.I., O'Hara, R.B., Scheiter, S., Huth, A., 2012. Connecting dynamic vegetation models to data - an inverse perspective. J. Biogeogr. 39, 2240–2252. https://doi.org/10.1111/j.1365-2699.2012.02745.x.

Lagarrigues, G., Jabot, F., Lafond, V., Courbaud, B., 2015. Approximate Bayesian computation to recalibrate individual-based models with population data: illustration with a forest simulation model. Ecol. Model 306, 278–286. https://doi.org/10.1016/j.ecolmodel.2014.09.023.

Hartig, Florian, Minunno, Francesco, Paul, Stefan, 2019. BayesianTools: General-Purpose MCMC and SMC Samplers and Tools for Bayesian Statistics. R package version 0.1.7. https://cran.r-project.org/package=BayesianTools.

Jasra, A., Stephens, D.A., Doucet, A., Tsagaris, T., 2011. Inference for lévy-driven stochastic volatility models via adaptive sequential Monte Carlo: lévy-driven stochastic volatility. Scand. J. Stat. 38, 1–22. https://doi.org/10.1111/j.1467-9469.2010.00723.x.

Jeffers, J.N.R, 1982. Modelling. Springer, Netherlands, Dordrecht. https://doi.org/10.1007/978-94-009-5968-2.

Jeremiah, E., Sisson, S., Marshall, L., Mehrotra, R., Sharma, A., 2011. Bayesian calibration and uncertainty analysis of hydrological models: a comparison of adaptive Metropolis and sequential Monte Carlo samplers. Water Resour. Res. 47 (7) https://doi.org/10.1029/2010WR010217.

Jeremiah, E., Sisson, S.A., Sharma, A., Marshall, L., 2012. Efficient hydrological model parameter optimization with Sequential Monte Carlo sampling. Environ. Model. Softw. 38, 283–295. https://doi.org/10.1016/j.envsoft.2012.07.001.

Laloy, E., Vrugt, J.A., 2012. High-dimensional posterior exploration of hydrologic models using multiple-try DREAM (ZS) and high-performance computing. Water Resour. Res. 48 https://doi.org/10.1029/2011WR010608.

Landsberg, J.J., Waring, R.H., 1997. A generalised model of forest productivity using simplified concepts of radiation-use efficiency, carbon balance and partitioning. For. Ecol. Manag. 95, 209–228. https://doi.org/10.1016/S0378-1127(97)00026-1.

Li, T., Sun, S., Sattar, T.P., Corchado, J.M., 2014. Fight sample degeneracy and impoverishment in particle filters: a review of intelligent approaches. Expert Syst. Appl. 41, 3944–3954. https://doi.org/10.1016/j.eswa.2013.12.031.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equation of state calculations by fast computing machines. J. Chem. Phys. 21, 1087–1092. https://doi.org/10.1063/1.1699114.

Minunno, F., Hartig, F., Trotsiuk, V. (2018) threePGN - A Fortran Implementation of the 3PGN Model for R. R package version 0.1.0, https://github.com/ForModLabUHel/threePGN-package.

Moradkhani, H., Hsu, K.-L., Gupta, H., Sorooshian, S., 2005. Uncertainty assessment of hydrologic model states and parameters: sequential data assimilation using the particle filter. Water Resour. Res. 41 https://doi.org/10.1029/2004WR003604.

Neal, R.M., 2001. Annealed importance sampling. Stat. Comput. 11, 125–139. https://doi.org/10.1023/A:1008923215028.

Oberpriller, J., Cameron, D.R., Dietze, M.C., & Hartig, F. (2021). Towards robust statistical inference for complex computer models. Ecol. Lett., in press.

Pontarp, M., Bunnefeld, L., Cabral, J.S., Etienne, R.S., Fritz, S.A., Gillespie, R., et al., 2019. The latitudinal diversity gradient: novel understanding through mechanistic eco-evolutionary models. Trends Ecol. Evol. (Amst.) 34, 11–223.

Schoups, G., Vrugt, J.A., 2010. A formal likelihood function for parameter and predictive inference of hydrologic models with correlated, heteroscedastic, and non-Gaussian errors. Water Resour. Res. 46 (10) https://doi.org/10.1029/2009WR008933.

Sisson, S.A., Fan, Y., Tanaka, M.M., 2007. Sequential Monte Carlo without likelihoods. Proc. Natl. Acad. Sci. 104, 1760–1765. https://doi.org/10.1073/pnas.0607208104.

Speekenbrink, M., 2016. A tutorial on particle filters. J. Math. Psychol. 73, 140–152. https://doi.org/10.1016/j.jmp.2016.05.006.

ter Braak, C.J.F., 2006. A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: easy Bayesian computing for real parameter spaces. Stat. Comput. 16, 239–249. https://doi.org/10.1007/s11222-006-8769-1.

ter Braak, C.J.F., Vrugt, J.A., 2008. Differential Evolution Markov Chain with snooker updater and fewer chains. Stat. Comput. 18, 435–446. https://doi.org/10.1007/s11222-008-9104-9.

... Trotsiuk, V., Hartig, F., Cailleret, M., Babst, F., Forrester, D.I., Baltensweiler, A., Schaub, M., 2020. Assessing the response of forest productivity to climate extremes in Switzerland using model–data fusion Glob. Chang Biol. 26 (4), 2463–2476.

Urban, M.C., Bocedi, G., Hendry, A.P., Mihoub, J.-.B., Pe, G., Singer, A., Travis, J.M.J., 2016. Improving the forecast for biodiversity under climate change. Science 353 (6304). https://doi.org/10.1126/science.aad8466.

Van Oijen, M., Rougier, J., Smith, R., 2005. Bayesian calibration of process-based forest models: bridging the gap between models and data. Tree Physiol 25, 915–927. https://doi.org/10.1093/treephys/25.7.915.

Vrugt, J.A., ter Braak, C.J.F., Diks, C.G.H., Robinson, B.A., Hyman, J.M., Higdon, D., 2009. Accelerating Markov Chain Monte Carlo Simulation by Differential Evolution with Self-Adaptive Randomized Subspace Sampling. Int. J. Nonlinear Sci. Numer. Simul. 10 https://doi.org/10.1515/IJNSNS.2009.10.3.273.

Vrugt, J.A., ter Braak, C.J.F., Diks, C.G.H., Schoups, G., 2013. Hydrologic data assimilation using particle Markov chain Monte Carlo simulation: theory, concepts and applications. Adv. Water Resour. 51, 457–478. https://doi.org/10.1016/j.advwatres.2012.04.002.

Zhu, G., Li, X., Ma, J., Wang, Y., Liu, S., Huang, C., Zhang, K., Hu, X., 2018. A new moving strategy for the sequential Monte Carlo approach in optimizing the hydrological model parameters. Adv. Water Resour. 114, 164–179. https://doi.org/10.1016/j.advwatres.2018.02.007.