Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

PEM-SMC: An algorithm for optimizing model parameters

Gaofeng Zhu [a], Qiang Chen [a], Xiangyu Yu [b], Cong Xu [a], Kun Zhang [c,d,*], Yunquan Wang [e,f,*], Wei Gong [g,h], Tao Che [i,*]

[a] *College of Earth and Environmental Sciences, Lanzhou University, Lanzhou, China*
[b] *Institute of Technology, Tianjin University of Finance and Economics, Tianjin, China*
[c] *School of Geospatial Engineering and Science, Sun Yat-sen University, Zhuhai, China*
[d] *Key Laboratory of Comprehensive Observation of Polar Environment (Sun Yat-sen University), Ministry of Education, China*
[e] *School of Environmental Studies, China University of Geosciences, Wuhan, China*
[f] *Laboratory of Basin Hydrology and Wetland Eco-restoration, China University of Geosciences, Wuhan, China*
[g] *State Key Laboratory of Earth Surface Processes and Resource Ecology, Faculty of Geographical Science, Beijing Normal University, Beijing, China*
[h] *Institute of Land Surface System and Sustainable Development, Faculty of Geographical Science, Beijing Normal University, Beijing, China*
[i] *Heihe Remote Sensing Experimental Research Station, Key Laboratory of Remote Sensing of Gansu Province, Northwest Institute of Eco-Environment and Resources, Chinese Academy of Sciences, Lanzhou, China*

ARTICLE INFO

Keywords:
Bayesian inference
MCMC
Sequential Monte-Carlo
Parameter optimization
Particle Evolution Metropolis Sequential Monte Carlo (PEM-SMC)

ABSTRACT

Bayesian inference is crucial for optimizing parameters in complex models, but often requires sampling due to high-dimensional, intractable posteriors. Beyond Markov-Chain Monte Carlo (MCMC) methods, Sequential Monte Carlo (SMC) algorithms offer an alternative. This paper introduces a Matlab toolbox for the Particle Evolution Metropolis Sequential Monte Carlo (PEM-SMC) algorithm, which combines the strengths of population-based MCMC and SMC. Two case studies – a complex multi-modal probability and a land surface model – demonstrate the toolbox's capabilities. This tool is valuable for Bayesian inference across fields like statistics, ecology, hydrology, and land surface processes.

Code metadata

Current code version	*V1*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2024-228
Permanent link to reproducible capsule	
Legal code license	*MIT*
Code versioning system used	*git*
Software code languages, tools and services used	*MATLAB*
Compilation requirements, operating environments and dependencies	*MATLAB 2018a (or later version)*
If available, link to developer documentation/manual	*For example:* https://github.com/kunlz/PEM-SMC/blob/main/README.md
Support email for questions	zhangkun3@mail.sysu.edu.cn

1. Introduction

A model is a simplified representation of reality with a set of unknown parameters. Based on Bayes' Theorem, Bayesian inference mathematically transforms prior probabilities into posterior distribution by considering the likelihood of the observed data, which offers powerful tool for integrating diverse data and managing uncertainties [1]. However, the equation for the posterior distribution is often complex, making direct sampling of parameters challenging [2]. Typically, these issues can be addressed using Markov chain Monte Carlo (MCMC) methods, with the Metropolis–Hastings algorithm being standard for generating parameter sequences [3]. To generate the candidate efficiently, many efforts have been devoted to designing the transition kernel, using either single or multiple chains running in parallel. However, MCMC does not reliably ensure that the sampling process converges to the target distribution.

* Corresponding authors.
E-mail addresses: zhangkun3@mail.sysu.edu.cn (K. Zhang), wangyq@cug.edu.cn (Y. Wang), chetao@lzb.ac.cn (T. Che).

https://doi.org/10.1016/j.simpa.2024.100728
Received 29 September 2024; Received in revised form 19 November 2024; Accepted 26 November 2024

Table 1
Comprehensive comparison of MCMC, SMC, and PEM-SMC.

	MCMC	SMC	PEM-SMC	References
Commonality	Using random sampling to approximate high-dimensional complex target distributions			Andrieu et al. [7];
Core idea	Iterative: Constructing a Markov chain to reach the stationary target distribution	Recursive: Updates particles with reweighting and resampling	Recursive: Updates particles with reweighting, resampling and mutation	Chopin [8]; Crisan and Doucet [9];
Sampling strategy	Accept-Reject: Relies on transition rules to accept or reject samples. (e.g., Metropolis–Hastings, Gibbs sampling)	Importance sampling: Uses importance weights to update and resample particles.	Hybrid sampling: Generates new particles under the SMC framework combined with an MCMC kernel	De Freitas [10];
Convergence	Slower: Chains converge slowly and require careful tuning	Faster: Resampling helps maintain convergence	Faster: Similar to SMC	Del Moral et al. [11];
Maintaining Diversity	Harder: Samples may become correlated, reducing diversity; using multiple chains help mitigate this issue	Easier: Resampling preserves diversity but risks degeneration.	Easier: Mutation operators prevent particle degeneration and preserve diversity	Hastings [3];
Applicability	Static: Well-suited for static, unimodal distributions but adaptable to more complex scenarios with advanced methods	Dynamic: Excels in dynamic systems or time-varying distributions (e.g., filtering problems)	Static: Ideal for resource-intensive models with multimodal distributions, high-dimensional static parameters	Kitagawa [12]; Metropolis et al. [13];
Parallelism	Harder: Stepwise dependency in Markov chain limits parallelism	Easier: Each particle's computation is independent	Easier: Similar to SMC and easily to achieve parallelism	Speich et al. [6]; Ter Braak and Vrugt [14];
Computation Efficiency	High efficient:Due to fewer chains but involves longer steps per chain	Efficient: Requires many particles and evolution steps, but parallelizability can reduce computational costs	Efficient: Similar to SMC	Vrugt et al. [15]; Zhu et al. [5]

Sequential Monte Carlo (SMC) methods offer an alternative, progressing parameters through a series of distributions from prior to posterior [4]. This method effectively explores multimodal distributions and represents the posterior using a particle set without relying on Markov properties. However, MCMC and SMC both have limitations, such as convergence issues and particle degeneracy, respectively. The future of Bayesian inference seems to be moving towards hybrid approaches that combine the strengths of MCMC and SMC with more modern computational techniques. In light of this, we developed the Particle Evolution Metropolis Sequential Monte Carlo (PEM-SMC) algorithm, combining both methods' strengths to enhance diversity and efficiency through iterative MCMC updates, proving superior to traditional approaches [5]. Table 1 compares MCMC, SMC, and PEM-SMC algorithm across key dimensions, including core ideas, sampling strategies, algorithm efficiency, and applicability. Recently, Speich et al. [6] reported that PEM-SMC outperform the state-of-the-art MCMC algorithms. Despite its advantages, PEM-SMC is not widely recognized. We introduce a MATLAB toolbox to facilitate its application, aiming to make this robust method accessible, detailed in this paper through various case studies.

2. A primer to PEM-SMC

The PEM-SMC algorithm iteratively evolves particles through a sequence of intermediate distributions, employing weighting, resampling, and moving steps (Fig. 1). In the weighting step, each particle receives a weight proportional to its density in the current distribution. Traditionally, resampling in SMC algorithms occurs only when the effective sample size falls below a specific threshold. However, we propose resampling at every iteration, regardless of sample size. This approach, while computationally demanding, offers two main advantages: it enhances the explorative capabilities of the particles, preventing degeneracy, and simplifies Monte Carlo estimates by equalizing particle weights post-resampling. We use a systematic resampling scheme in this study, though other methods like stratified, residual, and multinomial resampling can also be integrated into the algorithm.

3. Matlab toolbox of PEM-SMC

The PEM-SMC algorithm is implemented with Matlab version 2018a (or higher), and the source code is stored in GitHub repository. The PEM-SMC code can be executed from the Matlab prompt by the command:

$$[\mathrm{parameter\_iteration}] = \mathrm{PEM\_sampler}(\mathrm{Np}, \mathrm{S}, \mathrm{bound})$$

where Np is the number of particles in population; S is the number of maximum iteration; bound is a 2×d matrix that contains the lower and upper bound values of the d-dimensional parameters; parameter_iteration is a Np×d×S output matrix that contains the parameter values of each particle during the whole iterations. PEM_sampler uses five main functions to implement its various functionalities and generate samples from the desired distributions. Among them, the content of the target function needs to be written by the user and the call to this function is:

$$\mathrm{L} = \mathrm{target}(\mathrm{x})$$

where x is a 1× d parameter vector, and L is the returned log-density value of inputted parameters. The workflow of applying the PEM-SMC is presented in Fig. 2.

4. Numerical examples

4.1. Case study 1: a two-dimensional probability distribution with 20 modes

The first case study involves a two-dimensional normal mixture distribution taken from Liang and Wong [16]:

$$p(x) = \sum_{i=1}^{20} \frac{\omega_i}{2\pi\sigma_i} \exp\{-\frac{1}{2\sigma_i^2}(\mathbf{X}-\boldsymbol{\mu}_i)'(\mathbf{X}-\boldsymbol{\mu}_i)\} \tag{1}$$

where $\sigma_1 = \sigma_2 = \dots = \sigma_{20} = 0.1$; $\omega_1 = \omega_2 = \dots = \omega_{20} = 0.05$. Since most local modes are more than 15 standard deviations away from the nearest ones, this mixture distribution poses a serious challenge for sampling algorithms, and thus serves as a good test. We applied the PEM-SMC sampler to this problem, with set Np = 3000, S = 800, and bound=[−1, −1; 10, 10]. The target function for the case 1 can be found details in the repository of PEM-SMC in GitHub.

Fig. 3a showed that all the modes of the target distribution were successfully visited by the particles generated by the PEM-SMC sampler in the last iteration (S = 800). Fig. 3b shows the sample path of the last 400 iterations of a selected particle, which visited all the 20 modes frequently. Thus, it is evident that the PEM-SMC sampler has excellent global exploration ability.

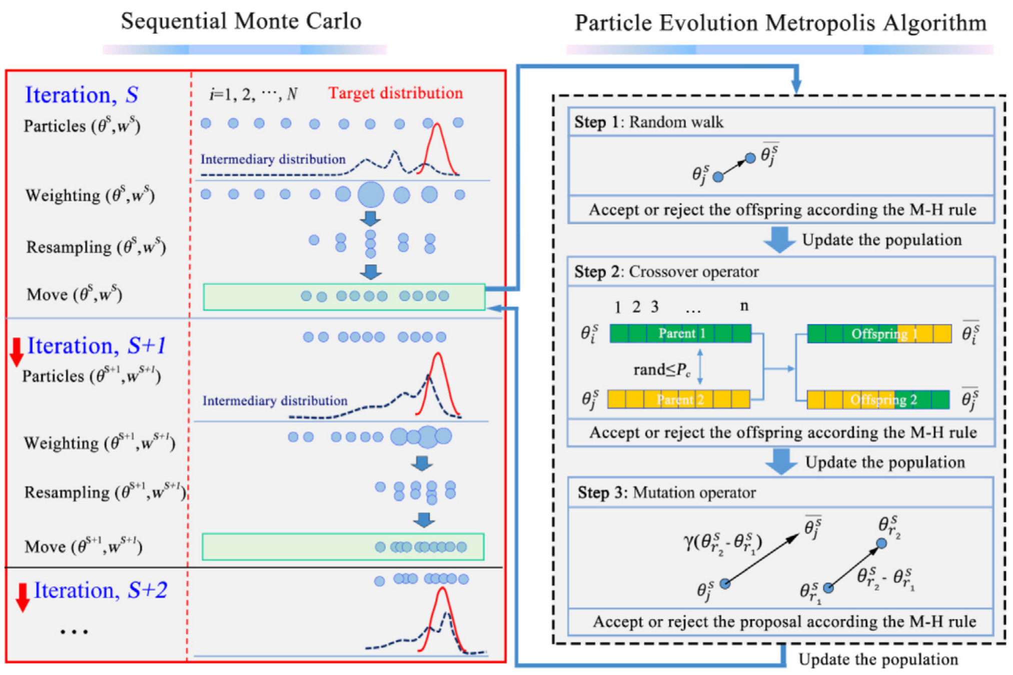
Sequential Monte Carlo
Particle Evolution Metropolis Algorithm
Iteration, S
Particles (θ^S,w^S)
i=1, 2, ⋯, N
Target distribution
Intermediary distribution
Weighting (θ^S,w^S)
Resampling (θ^S,w^S)
Move (θ^S,w^S)
Iteration, S+1
Particles (θ^{S+1},w^{S+1})
Weighting (θ^{S+1},w^{S+1})
Resampling (θ^{S+1},w^{S+1})
Move (θ^{S+1},w^{S+1})
Iteration, S+2
Step 1: Random walk
Accept or reject the offspring according the M-H rule
Update the population
Step 2: Crossover operator
Parent 1
Parent 2
Offspring 1
Offspring 2
rand≤P_c
Step 3: Mutation operator
$\gamma(\theta^S_{r_2}-\theta^S_{r_1})$
Accept or reject the proposal according the M-H rule
Update the population

Fig. 1. The basic sampling scheme of the PEM-SMC algorithm. In the moving step of SMC (left panel), the rand walk, crossover and mutation operators are employed to increase the diversities of particles (right panel).

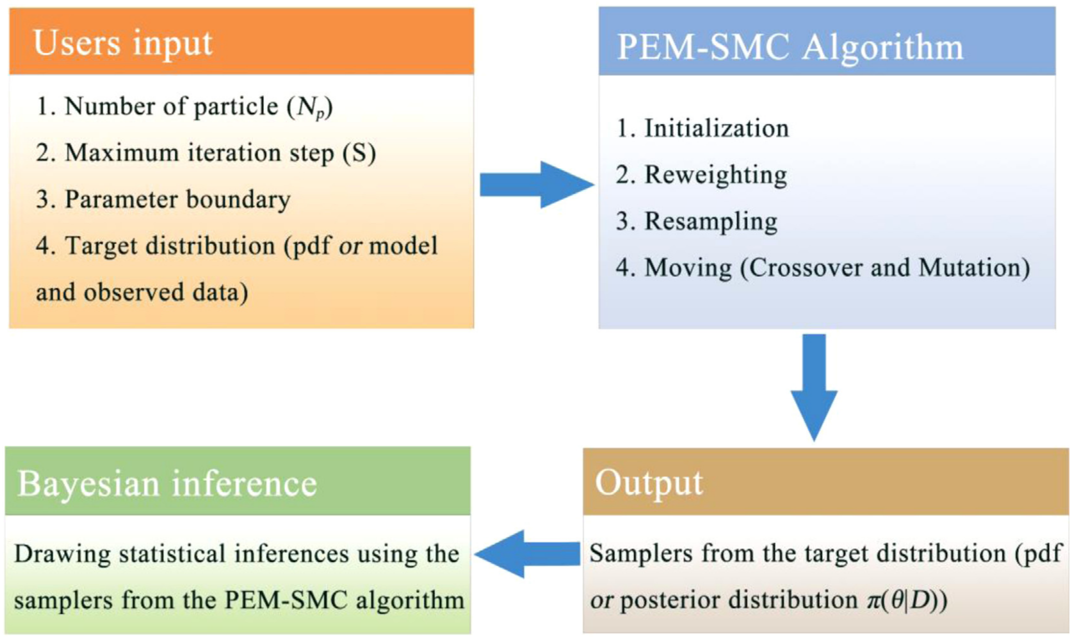
Users input
1. Number of particle (N_p)
2. Maximum iteration step (S)
3. Parameter boundary
4. Target distribution (pdf or model and observed data)
PEM-SMC Algorithm
1. Initialization
2. Reweighting
3. Resampling
4. Moving (Crossover and Mutation)
Output
Samplers from the target distribution (pdf or posterior distribution $\pi(\theta|D)$)
Bayesian inference
Drawing statistical inferences using the samplers from the PEM-SMC algorithm

Fig. 2. Inputs and outputs of the PEM-SMC algorithm, and modules contained in the algorithm. Only three input variables (Np, S, bound) and the target function are needed to be defined by users when applying the algorithm.

4.2. Case study 2: land surface model

The second case study explores the use of the PEM-SMC algorithm for parameter optimization in complex land surface models (LSMs), which are typically written in Fortran, C, or C++ rather than Matlab. This necessitates additional compiling steps when integrating the PEM-SMC algorithm with LSMs. We demonstrate this process using the Common Land Model (CoLM), a Fortran-based model extensively applied in simulating interactions between the land surface and atmosphere [17].

After generating the executable file (.exe), users can directly apply the PEM-SMC algorithm for LSM optimization. In this study, we used the latent heat flux (LE) observations in an evergreen needleleaf forest station (RU-FY2) from Fluxnet (https://fluxnet.org) to optimize the parameters of CoLM. Six parameters (*vmax25, α, m, ψ_d, B_d, N*) were selected by using the global sensitivity analyses. Users needed to define the Np, S, and bound, but to save computing time, Np and S were significantly reduced compared to previous cases. In the target function, the executable "run.exe" of CoLM simulated LE using parameters from input_step.txt, and the log-likelihood was calculated from the simulated results (model_data) and observations.

The particle evolutions of the six parameters were shown in Fig. 4. We can observe that all particles converged after 50 iterations, and the

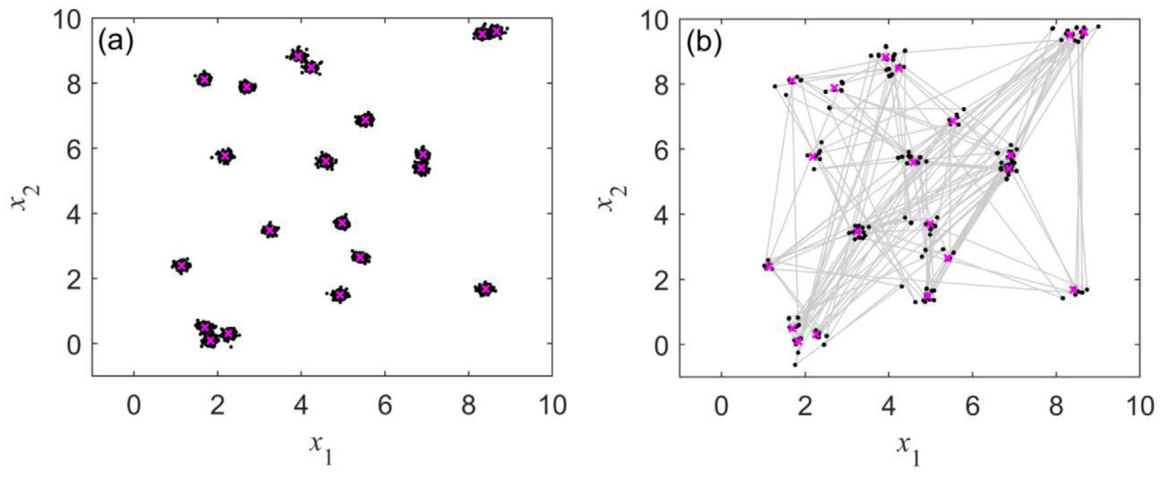

Fig. 3. (a) Samples generated from the last iteration of the PEM-SMC sampler. The actual values of local modes are represented by red cross ('×'), and the individual particles obtained by PEM-SMC sampler in the last iteration is coded with blue dot ('.'); (b) The sample path of the last 400 iterations of the selected particle generated by the PEM-SMC sampler.

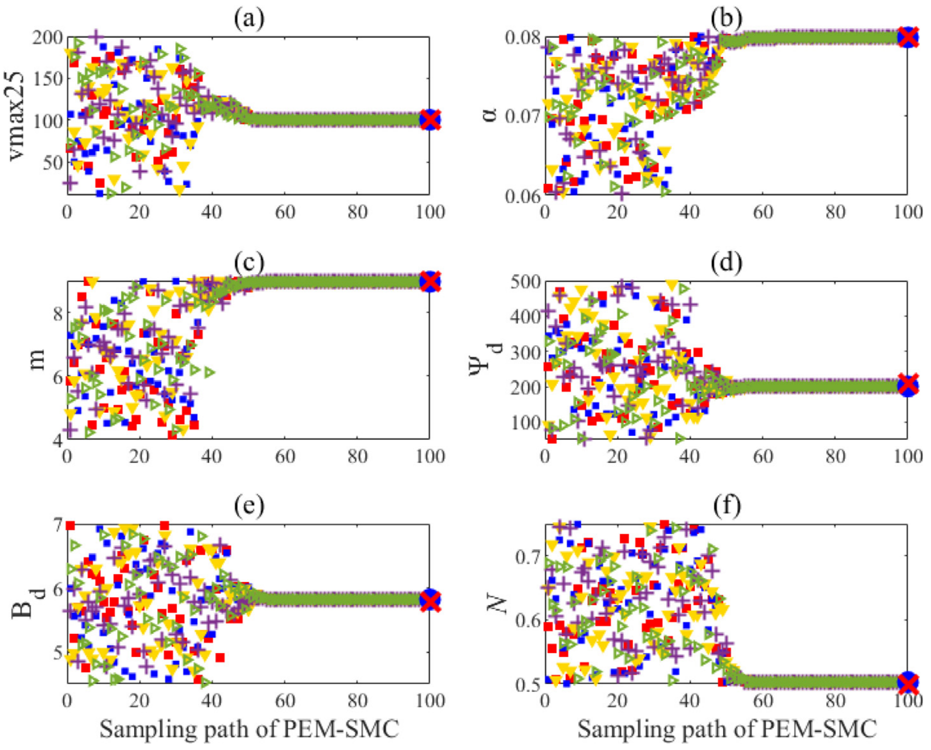

Fig. 4. Transitions of the sampled parameter values of (a) vmax25, (b) α, (c) m, (d) Ψd, (e) Bd, and (f) N. The cross symbol at the right-hand side of each plot indicates the actual (default) parameter values and the solid circles are the median value of the posterior distribution of the parameters. The five randomly selected particles are coded with different symbols and colors.

median values of the posterior parameter distributions match well with the true values. Therefore, we believe that the PEM-SMC algorithm is an effective tool in optimizing parameters of complex LSMs.

5. Impact overview

The PEM-SMC leverages a trio of transition operators (random walk, crossover, mutation) during the moving step to enhance the generation of new candidate particles, thereby boosting sampling efficiency. This approach, while computationally demanding, is particularly effective for complex models such as CoLM. Recent study [18] has tested the exclusive use of the mutation operator in the moving step, achieving over a 40% reduction in runtime without substantially compromising particle diversity. The mutation operator alone proves adequate for calibrating process-based models typically characterized by unimodal posterior distributions. However, for models exhibiting multimodal distributions or high sensitivity to parameter changes, the combined use of all three operators (random walk, crossover, mutation) is more advantageous. Specifically, the crossover operator facilitates particle escape from local optima, and the random walk operator is crucial for precise adjustments near optimal parameter values.

The implications of this algorithm extend beyond its immediate application, as demonstrated by case studies using our Matlab toolbox. This toolbox addresses posterior sampling challenges in diverse fields such as statistics, ecology, hydrology, and land surface modeling, showcasing its broad impact and utility in scientific research and model calibration.

6. Limitations and future work

A key challenge in employing the PEM-SMC algorithm is determining the optimal number of particles (Np) and the maximum number of iterations (S) to effectively transition from the initial to the final posterior distribution. These parameters affect the transition from initial to final posterior distributions, with a trade-off between stability and computational efficiency. A small S may cause particle degeneracy, while a large S results in unnecessary computations. Dynamic adjustment of S has been suggested to improve this balance.

Theoretically, Np should increase exponentially with the dimension of the model parameters (d). However, high Np values greatly increase computational time with little impact on accuracy. Practically, starting Np at 20 times d helps narrow the prior distribution, with adjustments to several hundreds or thousands over an appropriate prior interval to obtain precious posterior quantities of interest based on Monte Carlo estimate. These strategies aim to optimize computational demands and inference accuracy. Future work will explore dynamic adjustments of these parameters to further improve the computational efficiency of the PEM-SMC algorithm.

CRediT authorship contribution statement

Gaofeng Zhu: Writing – review & editing, Methodology, Formal analysis, Conceptualization. **Qiang Chen:** Writing – review & editing, Investigation. **Xiangyu Yu:** Investigation, Data curation. **Cong Xu:** Writing – review & editing, Methodology. **Kun Zhang:** Writing – review & editing, Methodology, Conceptualization. **Yunquan Wang:** Writing – review & editing, Formal analysis. **Wei Gong:** Writing – review & editing. **Tao Che:** Writing – review & editing, Funding acquisition, Conceptualization.

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

This work received funding from the National Natural Science Foundation of China (grant nos. 4212504 and 42171019) and the Key Science and Technology Foundation of Gansu Province (grant no. 23JRRA1025). We would like to express our sincere gratitude to the principal investigators and their teams for providing the site observation datasets used in this study. In addition, we acknowledge the support from the Supercomputing Center of Lanzhou University.

References

[1] J.S. Clark, A.E. Gelf, A future for models and data in environmental science, TRENDS Ecol. Evol. 21 (7) (2006) 375–380.
[2] R. van de Schoot, S. Depaoli, R. King, B. Kramer, K. Märtens, M.G. Tadesse, et al., Bayesian statistics and modelling, Nature Rev. Methods Primers 1 (1) (2021) 1.
[3] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika 57 (1970) 97–109.
[4] Y. Fan, D.S. Leslie, M.P. Wand, Generalised linear mixed model analysis via sequential Monte Carlo sampling, Electron. J. Stat. 2 (2008) 916–938.
[5] G. Zhu, X. Li, J. Ma, Y. Wang, S. Liu, C. Huang, K. Zhang, X. Hu, A new moving strategy for the sequential Monte Carlo approach in optimizing the hydrological model parameters, Adv. Water Resour. 114 (2018) 164–179.
[6] M. Speich, C.F. Dormann, F. Hartig, Sequential Monte-Carlo algorithms for Bayesian model calibration - A review and method comparison, Ecol. Model. 455 (2021) 109608.
[7] C. Andrieu, A. Doucet, R. Holenstein, Particle markov chain monte carlo methods, J. R. Stat. Soc. Ser. B Stat. Methodol. 72 (3) (2010) 269–342.
[8] N. Chopin, A sequential particle filter method for static models, Biometrika 89 (3) (2002) 539–552.
[9] D. Crisan, A. Doucet, A survey of convergence results on particle filtering methods for practitioners, IEEE Trans. Signal Process. 50 (3) (2002) 736–746.
[10] N. De Freitas, in: & N. J. Gordon A. Doucet (Ed.), Sequential Monte Carlo Methods in Practice (Vol. 1, No. 2), springer., New York, 2001, p. 2.
[11] P. Del Moral, A. Doucet, A. Jasra, Sequential monte carlo samplers, J. R. Stat. Soc. Ser. B Stat. Methodol. 68 (3) (2006) 411–436.
[12] G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, J. Comput. Graph. Stat. 5 (1) (1996) 1–25.
[13] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (6) (1953) 1087–1092.
[14] C.J. Ter Braak, J.A. Vrugt, Differential evolution Markov chain with snooker updater and fewer chains, Stat. Comput. 18 (2008) 435–446.
[15] J.A. Vrugt, C.J. ter Braak, C.G. Diks, G. Schoups, Hydrologic data assimilation using particle Markov chain Monte Carlo simulation: Theory, concepts and applications, Adv. Water Resour. 51 (2013) 457–478.
[16] F. Liang, W.H. Wong, Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models, J. Amer. Statist. Assoc. 96 (2001) 653–666.
[17] Y. Dai, X. Zeng, R.E. Dickinson, I. Baker, G.B. Bonan, M.G. Bosilovich, et al., The common land model, Bull. Am. Meteorol. Soc. 84 (8) (2003) 1013–1024.
[18] C. Xu, G.F. Zhu, Y. Zhang, K. Zhang, Comparing the Impacts of Single- and Multi-Objective Optimization on the Parameter Estimation and Performance of a Land Surface Model. Hydrology and Earth System Sciences Discussions, 2024.