Statistica Sinica **34** (2024), 1071-1091
doi:https://doi.org/10.5705/ss.202022.0172

A LANGEVINIZED ENSEMBLE KALMAN FILTER FOR LARGE-SCALE DYNAMIC LEARNING

Peiyi Zhang, Qifan Song and Faming Liang*

Purdue University

Abstract: The ensemble Kalman filter (EnKF) performs well in terms of data assimilation in atmospheric and oceanic sciences. However, it fails to converge to the correct filtering distribution, which precludes its use for uncertainty quantification in dynamic systems. Thus, we reformulate the EnKF under the framework of Langevin dynamics, yielding a new particle filtering algorithm, which we call the Langevinized EnKF (LEnKF). The LEnKF inherits the forecast-analysis procedure from the EnKF, and uses mini-batch data from stochastic gradient Langevin dynamics (SGLD). We prove that the LEnKF is a sequential preconditioned SGLD sampler, like the EnKF, but with its execution accelerated by the forecast-analysis procedure. Furthermore, the LEnKF converges to the correct filtering distribution in terms of the 2-Wasserstein distance as the number of iterations per i stage increases. We demonstrate the performance of the LEnKF using a variety of examples. The LEnKF is not only scalable with respect to the state dimension and the sample size, but also tends to be immune to sample degeneracy for long-series dynamic data.

Key words and phrases: Data assimilation, inverse problem, state space model, stochastic gradient Markov chain Monte Carlo, uncertainty quantification.

1. Introduction

The integration of computer technology into science and daily life has enabled scientists to collect massive volumes of data, such as climate data, high-throughput biological assay data, and website transaction logs. To address the computational difficulties that arise in Bayesian analyses of big data, several scalable MCMC algorithms have been developed, including the stochastic gradient MCMC algorithms (Welling and Teh (2011); Ding et al. (2014); Chen, Fox and Guestrin (2014); Li et al. (2016); Ma, Chen and Fox (2015); Nemeth and Fearnhead (2021)), split-and-merge algorithms (Scott et al. (2016); Li, Srivastava and Dunson (2017); Srivastava, Li and Dunson (2018)), mini-batch Metropolis–Hastings algorithms (Chen et al. (2018); Maclaurin and Adams (2014); Bardenet, Doucet and Holmes (2017)), and nonreversible Markov process-based algorithms (Bierkens, Fearnhead and Roberts (2019); Bouchard-Côté, Vollmer and Doucet (2018)).

*Corresponding author.

Although scalable MCMC algorithms perform well in Bayesian learning with static data, none can be applied directly to dynamic data. In the literature, learning based on static data is often referred to as static or offline learning, and that based on dynamic data is called as dynamic or online learning. Dynamic learning is important and challenging, because dynamic data collection is general, heterogeneous, and messy. Note that classical sequential Monte Carlo or particle filter algorithms (e.g., see Liu and Chen (1998) and Doucet, de Freitas and Gordon (2001)) lack the scalability necessary to handle large-scale dynamic data, where we wish to use all available data at each processing step. The ensemble Kalman filter (EnKF) (Evensen (1994)) is efficient for high-dimensional data-assimilation problems (e.g., Evensen and van Leeuween (1996); Aanonsen et al. (2009); Houtekamer and Mitchell (2011)), but fails to converge to the correct filtering distribution for general nonlinear dynamic systems (Law, Tembine and Tempone (2016)). Thus, there is a need for statistical methods for Bayesian on-line learning based on large-scale dynamic data.

In this study, we conduct Bayesian online learning for the following dynamic system:

$$\begin{aligned} x_t &= g(x_{t-1}) + u_t, \quad u_t \sim N(0, U_t), \\ y_t &= H_t x_t + \eta_t, \quad \eta_t \sim N(0, \Gamma_t), \end{aligned} \tag{1.1}$$

for stages $t = 1, 2, \ldots, T$, where $x_t \in \mathbb{R}^p$ and $y_t \in \mathbb{R}^{N_t}$ denote the state and observations, respectively, at stage t. The dimension p, number of stages T, and sample size N_t are all assumed to be on a large scale. For the dynamic system (1.1), the top equation is called the state evolution equation, where $g(\cdot)$ is called state propagator, and can be nonlinear. The bottom equation is called the measurement equation, where the propagator H_t relates the state variable to the measurement variable, and yields the expected value of the prediction, given the state and the parameters. The dynamic system (1.1) is of central importance. First, it models data-assimilation problems that have linear measurement equations. Second, many other problems, such as the inverse problem and data-assimilation problems that have nonlinear measurement equations can be converted to (1.1) by using appropriate transformations, as discussed in Section 3.1 and Section 5. For simplicity, we assume that both the model error u_t and the observation error η_t are zero-mean Gaussian random variables, and that the covariance matrices U_t and Γ_t and the propagators $g(\cdot)$ and H_t are all fully specified; that is, they contain no unknown parameters. How to extend our results to problems with non-Gaussian observations and/or unknown parameters is discussed in Section 5.

Throughout this paper, we let t be an index of the stage of the dynamic system, let $f(y_t|x_t)$ denote the likelihood function of y_t, let $\pi(x_t|y_{1:t})$ denote the filtering distribution at stage t given the data $y_{1:t} = \{y_1, y_2, \ldots, y_t\}$, and let

$\pi(x_t|y_{1:t-1}) = \int \pi(x_t|x_{t-1})\pi(x_{t-1}|y_{1:t-1})dx_{t-1}$ denote the predictive distribution of x_t given $y_{1:t-1}$.

Our results contribute to the literature in two ways. First, We develop a new particle filter, the so-called Langevinized EnKF (LEnKF), by reformulating the EnKF under the framework of Langevin dynamics. The LEnKF is not only scalable with respect to the state dimension and sample size, but also tends to be immune to the sample degeneracy problem often encountered by conventional particle filters. The LEnKF works well in a big data scenario in which the stage number T, state dimension p, and sample sizes N_t are all large scale. Second, we prove that the LEnKF is a sequential preconditioned SGLD sampler; however, its execution is accelerated by the forecast-analysis procedure, and it converges to the correct filtering distribution in terms of 2-Wasserstein distance as the number of iterations per stage increases. The LEnKF can be used efficiently for uncertainty quantification in large-scale dynamic systems. We illustrate the performance of the LEnKF using several examples, including the Lorenz-96 model (Lorenz (1996)) and the long short-term memory (LSTM) model (Hochreiter and Schmidhuber (1997)). To conserve space, the latter is presented in the Supplementary Material. To the best of our knowledge, this work represents the first development of a scalable particle filter under a rigorous probabilistic framework.

The remainder of this paper is organized as follows. Section 2 provides a brief review of the EnKF, and explains its scalability with respect to the state dimension. Section 3 describes the LEnKF for dynamic learning and studies its convergence. In Sections 4, we demonstrate the performance of the LEnKF using a variable selection example and the Lorenz-96 model. Section 5 discusses possible extensions of the LEnKF. Section 6 concludes the paper.

2. Why is the EnKF Efficient for High-Dimensional Problems?

Consider the dynamic system (1.1). To estimate the state variables $x_1, x_2, \ldots, x_T$, Evensen (1994) proposed the EnKF algorithm, as described in Algorithm 1. If $U_t, \Gamma_t, g(\cdot)$, and H_t contain unknown parameters, we can use the state augmentation method (Anderson (2001); Baek et al. (2006); Gillijns and De Moor (2007)), where the state vector is augmented with unknown parameters, and the state and parameters are estimated simultaneously.

The EnKF has two attractive features that help it perform well in high-dimensional data-assimilation problems, such as those encountered in reservoir modeling (Aanonsen et al. (2009)), oceanography (Evensen and van Leeuween (1996)), and weather forecasting (Houtekamer and Mitchell (2011)). First, the EnKF approximates each filtering distribution $\pi(x_t|y_{1:t})$ using an ensemble of particles. Because the ensemble size m is typically much smaller than p, it leads to better dimension reduction and computational feasibility than that of the

Algorithm 1 EnKF Algorithm.

Initialization: Initialize an ensemble $\{x_0^{a,1}, x_0^{a,2}, \ldots, x_0^{a,m}\}$ of size m.
for $t = 1$ **to** T **do**
- **(i) Forecast:** For $i = 1, 2, \ldots, m$, draw $u_t^i \sim N(0, U_t)$ and set $x_t^{f,i} = g(x_{t-1}^{a,i}) + u_t^i$; calculate the sample covariance matrix of $x_t^{f,1}, \ldots, x_t^{f,m}$ and denote it by C_t.
- **(ii) Analysis:** For $i = 1, 2, \ldots, m$, draw $\eta_t^i \sim N(0, \Gamma_t)$ and set $x_t^{a,i} = x_t^{f,i} + \hat{K}_t(y_t - H_t x_t^{f,i} - \eta_t^i) \triangleq x_t^{f,i} + \hat{K}_t(y_t - y_t^{f,i})$, where $\hat{K}_t = C_t H_t^T (H_t C_t H_t^T + \Gamma_t)^{-1}$ forms an estimator for the Kalman gain matrix $K_t = S_t H_t^T \ (H_t S_t H_t^T + \Gamma_t)^{-1}$ and S_t denotes the covariance matrix of x_t^f.

end for

Kalman filter; see, for example, Shumway and Stoffer (2006). In particular, it approximates S_t using C_t, so the storage for the matrix C_t is replaced with particles, and is much reduced. Second, by generating particles from filtering distributions, it avoids covariance matrix decomposition. It is known that an LU-decomposition of a covariance matrix has computational complexity $O(p^3)$. Instead, the EnKF employs a forecast-analysis procedure to generate particles, which has computational complexity $O(\max\{p^2 N_t, N_t^3\} + mpN_t)$, for m particles at stage t. That is, the forecast-analysis procedure reduces the computational complexity of the particle generation when m and N_t are smaller than p. This explains why the EnKF is so efficient for high-dimensional problems. On the other hand, this also implies that the EnKF may be inefficient when N_t is large.

Despite its success in dealing with high-dimensional dynamic systems, the performance of the EnKF is sub-optimal. As shown by Law, Tembine and Tempone (2016), it converges only to a mean-field filter, which provides the optimal linear estimator of the conditional mean, but not the filtering distribution, except for linear systems in the large-sample limit. Similar results can be found in Le Gland, Monbet and Tran (2009), Bergou, Gratton and Mandel (2019), and Kwiatkowski and Mandel (2015).

Using the state augmentation approach (Iglesias, Law and Stuart (2013)), the EnKF can also be used to solve the inverse problem, that is, determining the parameter x for the system

$$y = \mathcal{G}(x) + \eta, \tag{2.1}$$

where $\mathcal{G}(\cdot)$ is the forward response operator mapping the unknown parameter x to the space of observations, $\eta \sim N(0, \Gamma)$ is Gaussian random noise, and y denotes the observed data. However, as mentioned previously, the EnKF does not converge to the filtering distribution; thus, we cannot approximate the posterior distribution $\pi(x|y)$ well using this approach. Numerically, Ernst, Sprungk and Starkloff (2015) show that for nonlinear inverse problems, the large-sample limit does not lead to a good approximation of the posterior distribution.

3. Langevinized Ensemble Kalman Filter

To motivate the development of the LEnKF, we first consider a linear inverse problem, and then extend it to the data-assimilation problem (1.1), and others. Note that, as implied by (1.1), estimating x_t at each individual stage is essentially a linear inverse problem, but with some "prior" information passed on from the preceding stage.

3.1. Linear inverse problem

Consider a Bayesian inverse problem for the regression

$$y = Hx + \eta, \tag{3.1}$$

where H is a known matrix, $\eta \sim N(0, \Gamma)$ for some covariance matrix Γ, $y \in \mathbb{R}^N$, and $x \in \mathbb{R}^p$ is an unknown continuous parameter vector. To accommodate the case that the sample size N is extremely large, we assume y can be partitioned into $B = N/n$ independent and identically distributed (i.i.d.) blocks $\{\tilde{y}_1, \ldots, \tilde{y}_B\}$, where each block is of size n and has a positives-definite covariance matrix V, such that $\Gamma = \text{diag}[V, \ldots, V]$. Note that this is a trivial assumption for independent samples, as considered here.

Let $\pi(x)$ denote the prior density function of x, which is assumed to be differentiable with respect to x. Let $\pi(x|y)$ denote the posterior distribution. To develop a scalable algorithm for simulating from $\pi(x|y)$ under the scenario that both N and p are large, we reformulate the model (3.1) as a state-space model using subsampling and Langevin diffusion:

$$\begin{aligned} x_t &= x_{t-1} + \epsilon_t \frac{n}{2N} \nabla \log \pi(x_{t-1}) + w_t, \\ y_t &= H_t x_t + v_t, \end{aligned} \tag{3.2}$$

where $w_t \sim N(0, (n/N)\epsilon_t I_p) = N(0, (n/N)Q_t)$, that is, $Q_t = \epsilon_t I_p$, y_t denotes a block drawn randomly from $\{\tilde{y}_1, \ldots, \tilde{y}_B\}$, $v_t \sim N(0, V)$, and H_t is a submatrix of H extracted using the corresponding y_t. In the state-space model, at each stage t, the state (i.e., the parameters of model (3.1)) evolves according to an Euler-discretized Langevin equation of the prior distribution, and the measurement equation varies with subsampling.

To simulate from the dynamic system (3.2), we propose Algorithm 2, which uses both subsampling and the forecast-analysis procedure, and is thus scalable with respect to both the sample size N and the state dimension p. Theorem 1 shows that the proposed algorithm is a parallel preconditioned SGLD algorithm; the proof is given in the Supplementary Material. Then, following from the general recipe of stochastic gradient MCMC (Ma, Chen and Fox (2015)), each chain of the algorithm converges to the target posterior $\pi(x|y)$ as $t \to \infty$, provided that $\epsilon_t \to 0$ as $t \to \infty$. As mentioned in Remark S1, the convergence of the

Algorithm 2 LEnKF for Linear Inverse Problems.

Initialization: Set $t = 0$ and initialize an ensemble $\{x_0^{a,1}, x_0^{a,2}, \ldots, x_0^{a,m}\}$ of size m.
for $t = 1$ **to** T **do**
(i) Subsampling: Draw y_t from $\{\tilde{y}_1, \ldots, \tilde{y}_B\}$. Set $Q_t = \epsilon_t I_p$, $R_t = 2V$, and the Kalman gain matrix $K_t = Q_t H_t^T (H_t Q_t H_t^T + R_t)^{-1}$.
for $i = 1$ **to** m **do**
(ii) Forecast: Draw $w_t^i \sim N_p(0, (n/N)Q_t)$ and set

$$x_t^{f,i} = x_{t-1}^{a,i} + \epsilon_t \frac{n}{2N} \nabla \log \pi(x_{t-1}^{a,i}) + w_t^i. \tag{3.4}$$

(iii) Analysis: Draw $v_t^i \sim N_n(0, (n/N)R_t)$ and set

$$x_t^{a,i} = x_t^{f,i} + K_t(y_t - H_t x_t^{f,i} - v_t^i) \triangleq x_t^{f,i} + K_t(y_t - y_t^{f,i}). \tag{3.5}$$

end for
end for

algorithm (measured in terms of 2-Wasserstein distance) also follows directly from Corollary S1.

Theorem 1. *For Algorithm* 2, *if* V *is positive definite, then the algorithm reduces to a parallel preconditioned SGLD algorithm that converges to the target posterior distribution* $\pi(x|y)$ *as* $t \to \infty$, *provided* $\epsilon_t \to 0$ *as* $t \to \infty$; *that is, for each chain* $i \in \{1, 2, \ldots, m\}$,

$$x_t^{a,i} = x_{t-1}^{a,i} + \frac{\epsilon_t}{2} \Sigma_t \widehat{\nabla} \log \pi(x_{t-1}^{a,i}|y) + e_t, \tag{3.3}$$

where $\Sigma_t = (n/N)(I - K_t H_t)$ *is a constant matrix of* x, e_t *is a zero-mean Gaussian random error with covariance* $Var(e_t) = \epsilon_t \Sigma_t$, *and* $\widehat{\nabla} \log \pi(x_{t-1}^{a,i}|y) = (N/n) H_t^T V_t^{-1}(y_t - H_t x_{t-1}^{a,i}) + \nabla \log \pi(x_{t-1}^{a,i})$ *represents an unbiased estimate of* $\nabla \log \pi(x_{t-1}^{a,i}|y)$.

The major advantage of such a reformulation from an inverse problem to a state-space model lies in the computation. For a high-dimensional problem, suppose we set the mini-batch size n much smaller than p; then, the computational complexity of LEnKF for T iterations is $O((n^2 p + mnp)T)$. In contrast, if (3.3) is simulated directly as a preconditioned algorithm for the original inverse problem, the computational complexity is $O((p^3 + np^2)mT)$, where $O(p^3)$ and $O(np^2)$ represent the costs of generating e_k and computing $\Sigma_k \widehat{\nabla} \log \pi(x_{k-1}^a|\boldsymbol{y})$, respectively. The former requires an LU-decomposition of Σ_t, which has computational complexity $O(p^3)$. The LEnKF avoids this problem by using a forecast-analysis procedure, making it scalable with respect to the state dimension p.

Note that the computational complexity of LEnKF is the same as that of the parallel SGLD algorithm (Welling and Teh (2011)). The latter algorithm consists

of m chains and each chain evolves via the iteration

$$x_t^i = x_{t-1}^i + \frac{\epsilon_t}{2}\widehat{\nabla}_x \log \pi(x_{t-1}^i|y) + \tilde{e}_t,\ i = 1, 2, \ldots, m,$$

where $\widehat{\nabla}_x \log \pi(x_{t-1}^i|y) = (N/n)H_t^T V^{-1}(y_t - H_t x_{t-1}^i) + \nabla \log \pi(x_{t-1}^i)$, as defined in (3.3), $\tilde{e}_t \sim N(0, \epsilon_t I_p)$, and, at each iteration t, all chains are updated based on the same mini-batch data y_t. As implied by Theorem 1 of Li et al. (2016), the LEnKF converges more quickly than the parallel SGLD does, because all eigenvalues of the preconditioner Σ_t can be much less than one by noting that $\Sigma_t = (n/N)(I - K_t H_t) = (n/N)(I - \epsilon_t H_t^T(\epsilon_t H_t H_t^T + 2V)^{-1} H_t)$. This is illustrated in Figure 2, and in Figure S1 and Figure S2 (in the Supplementary Material).

3.2. Data assimilation with linear measurement equations

Consider the dynamic system (1.1). Similarly to the linear inverse problem, we assume that at each stage t, y_t can be partitioned into $B_t = N_t/n_t$ i.i.d. blocks $\{\tilde{y}_{t,1}, \ldots, \tilde{y}_{t,B_t}\}$, where each block has size n_t, and $\tilde{y}_{t,k} = H_{t,k} x_t + v_{t,k}$, for $k = 1, 2, \ldots, B_t$; N_t is the total number of observations at stage t, $v_{t,k} \sim N(0, V_t)$, for all k; and $v_{t,k}$ are mutually independent; that is, $\Gamma_t = \text{diag}[V_t, \ldots, V_t]$. Again, we assume that V_t is positive definite. Let $y_{t,k}$ denote a block of n_t observations drawn randomly from the set $\{\tilde{y}_{t,1}, \ldots, \tilde{y}_{t,B_t}\}$.

To motivate the development of the algorithm, we first consider the Bayesian formula

$$\pi(x_t|y_{1:t}) = \frac{f(y_t|x_t)\pi(x_t|y_{1:t-1})}{\int f(y_t|x_t)\pi(x_t|y_{1:t-1})dx_t}, \tag{3.6}$$

which suggests that in order to get the filtering distribution $\pi(x_t|y_{1:t})$, we first need to use the predictive distribution $\pi(x_t|y_{1:t-1})$ as the prior at stage t. To estimate the gradient $\nabla_{x_t} \log \pi(x_t|y_{1:t-1})$, we employ the following identity established in Song et al. (2020):

$$\nabla_\beta \log \pi(\beta \mid D) = \int \nabla_\beta \log \pi(\beta \mid \gamma, D)\pi(\gamma \mid \beta, D)d\gamma, \tag{3.7}$$

where D denotes data, and β and γ denote two parameters of a posterior distribution $\pi(\beta, \gamma|D)$. By this identity, we have

$$\begin{aligned}
\nabla_{x_t} \log \pi(x_t|y_{1:t-1}) &= \int \nabla_{x_t} \log \pi(x_t|x_{t-1}, y_{1:t-1})\pi(x_{t-1}|x_t, y_{1:t-1})dx_{t-1} \\
&= \int \nabla_{x_t} \log \pi(x_t|x_{t-1}) \frac{\pi(x_{t-1}|x_t, y_{1:t-1})}{\pi(x_{t-1}|y_{1:t-1})} \pi(x_{t-1}|y_{1:t-1})dx_{t-1} \\
&= \int \nabla_{x_t} \log \pi(x_t|x_{t-1})\omega(x_{t-1}|x_t)\pi(x_{t-1}|y_{1:t-1})dx_{t-1},
\end{aligned} \tag{3.8}$$

where $\omega(x_{t-1}|x_t) = \pi(x_{t-1}|x_t, y_{1:t-1})/\pi(x_{t-1}|y_{1:t-1}) = \pi(x_t|x_{t-1})/\pi(x_t|y_{1:t-1})$

$\propto \pi(x_t|x_{t-1})$, because $\pi(x_t|y_{1:t-1})$ is a constant with respect to x_{t-1}, given the particle x_t and the data $y_{1:t-1}$. Therefore, given a set of samples $\mathcal{X}_{t-1} = \{x_{t-1,1}, x_{t-1,2}, \ldots, x_{t-1,m'}\}$ drawn from the filtering distribution $\pi(x_{t-1}|y_{1:t-1})$, we can use an importance resampling procedure to draw a sample from $\pi(x_{t-1}|x_t, y_{1:t-1})$. The importance resampling procedure can be executed very fast, because calculating the importance weight $\omega(x_{t-1}|x_t)$ does not involve any data.

Using the above formulae, we can construct a dynamic system, similar to (3.2), for the data-assimilation problem (1.1) at stage t, as

$$\begin{aligned} x_{t,k} &= x_{t,k-1} - \epsilon_t \frac{n_t}{2N_t} U_t^{-1}(x_{t,k-1} - g(\tilde{x}_{t-1,k-1})) + w_{t,k}, \\ y_{t,k} &= H_{t,k} x_{t,k} + v_{t,k}, \end{aligned} \tag{3.9}$$

for $k = 1, 2, \ldots$, where $x_{t,0} = g(x_{t-1}) + u_t$, $\tilde{x}_{t-1,k-1}$ represents a sample of $\pi(x_{t-1}|x_{t,k-1}, y_{1:t-1})$ and is drawn from $\mathcal{X}_{t-1}$ using an importance resampling procedure, $w_{t,k} \sim N(0, (n_t/N_t)\epsilon_{t,k} I_p)$, $Q_{t,k} = \epsilon_{t,k} I_p$, and p is the dimension of x_t. Applying Algorithm 2 to (3.9) at each stage t leads to Algorithm 3. The convergence theory of the algorithm is studied in Theorem 2, the proof of which is given in the Supplementary Material.

Theorem 2. *We consider a dynamic system with $t = 1, 2, \ldots, T$ stages. Let $\pi_t = \pi(x_t|y_{1:t})$ denote the filtering distribution at stage t. Suppose Assumptions* S1– S8 *(given in the Supplementary Material) hold, and N_t is larger than a certain threshold. If $\epsilon_{t,k} \propto (1/n_t^2 \log \mathcal{K} k^{-\varpi})$, for some $\varpi \in (0, 1)$ and any $k \in \{1, 2, \ldots, \mathcal{K}\}$, then uniformly with dominating probability, for any $t \in \{1, 2, \ldots, T\}$, $x_{t,\mathcal{K}}^{a,i}$ follows a probability law $\tilde{\pi}_t$ and $\lim_{\mathcal{K}\to\infty} W_2(\tilde{\pi}_t, \pi_t) = 0$, where $W_2(\cdot, \cdot)$ denotes the 2-Wasserstein distance between two distributions.*

The following remarks relate to Algorithm 3 and Theorem 2.

Remark 1. (On asymptotic regime). Theorem 2 studies the convergence of Algorithm 3 under the asymptotic regime that the number of iterations at each stage (i.e., $\mathcal{K}$) diverges. Such a result is similar to the convergence theory of a sequential Monte Carlo algorithm as the number of particles increases to infinity (see, e.g., Beskos et al. (2016); Crisan and Doucet (2000)). Under this asymptotic regime, we provide a rigorous study for the convergence of the algorithm; in particular, we account for the approximation error of $\tilde{\pi}_t$ to π_t for each stage t in establishing the convergence of the algorithm. See equations (S2.23), (S2.25) and (S2.27) of the Supplementary Material for further detail.

Remark 2. (On sample degeneracy). It is known that sample degeneracy is an inherent problem with sequential importance sampling (Cappé et al. (2004)), especially when the dimension of the system is high. When it occurs, the importance weights concentrate on a few samples, the effective sample size is low, and the resulting importance sampling estimate is heavily biased. Fortunately,

Algorithm 3 LEnKF for Data-Assimilation Problems.

Initialization: Initialize an ensemble $\{x_{1,0}^{a,1}, x_{1,0}^{a,2}, \ldots, x_{1,0}^{a,m}\}$ of size m by drawing from the prior distribution $\pi(x_1)$. Set $\mathcal{X}_t = \emptyset$, for $t = 1, 2, \ldots, T$; set the learning rate sequence $\{\epsilon_{t,k} : t = 1, 2, \ldots, T,\ k = 1, 2, \ldots, \mathcal{K}\}$, where $\mathcal{K}$ denotes the number of iterations performed at each stage; and set k_0 as the common burn-in period of each stage.

for $t = 1$ **to** T **do**

  for $k = 1$ **to** $\mathcal{K}$ **do**

  (i) Subsampling: Draw a mini-batch sample $y_{t,k}$ from the set $\{\tilde{y}_{t,1}, \ldots, \tilde{y}_{t,B_t}\}$. Set $Q_{t,k} = \epsilon_{t,k} I_p$, $R_t = 2V_t$, and the Kalman gain matrix $K_{t,k} = Q_{t,k} H_{t,k}^T (H_{t,k} Q_{t,k} H_{t,k}^T + R_t)^{-1}$.

  for $i = 1$ **to** m **do**

  (ii) Importance resampling: If $t > 1$, calculate importance weights $\omega_{t,k-1,j}^i = \pi(x_{t,k-1}^{a,i} | x_{t-1,j}) = \phi(x_{t,k-1}^{a,i}; g(x_{t-1,j}), U_t)$, for $j = 1, 2, \ldots, |\mathcal{X}_{t-1}|$, where $\phi(\cdot)$ denotes a Gaussian density, and $x_{t-1,j} \in \mathcal{X}_{t-1}$ denotes the jth sample in $\mathcal{X}_{t-1}$; if $k = 1$, set $x_{t,0}^{a,i} = g(x_{t-1,\mathcal{K}}^{a,i}) + u_t^{a,i}$ and $u_t^{a,i} \sim N(0, U_t)$. Resample $s \in \{1, 2, \ldots, |\mathcal{X}_{t-1}|\}$ with a probability $\propto \omega_{t,k-1,s}^i$, i.e., $P(S_{t,k,i} = s) = \omega_{t,k-1,s}^i / \sum_{j=1}^{|\mathcal{X}_{t-1}|} \omega_{t,k-1,j}^i$, and denote the sample drawn from $\mathcal{X}_{t-1}$ by $\tilde{x}_{t-1,k-1}^i$.

  (iii) Forecast: Draw $w_{t,k}^i \sim N_p(0, (n/N) Q_{t,k})$. If $t = 1$, set $x_{t,k}^{f,i} = x_{t,k-1}^{a,i} - \epsilon_{t,k}(n_t/2N_t) \nabla \log \pi(x_{t,k-1}^{a,i}) + w_{t,k}^i$, where $\pi(\cdot)$ denotes the prior distribution of x_1; otherwise, set $x_{t,k}^{f,i} = x_{t,k-1}^{a,i} - \epsilon_{t,k}(n_t/2N_t) U_t^{-1} (x_{t,k-1}^{a,i} - g(\tilde{x}_{t-1,k-1}^i)) + w_{t,k}^i$.

  (iv) Analysis: Draw $v_{t,k}^i \sim N_n(0, (n/N) R_t)$ and set

$$\begin{aligned} x_{t,k}^{a,i} &= x_{t,k}^{f,i} + K_{t,k}(y_{t,k} - H_{t,k} x_{t,k}^{f,i} - v_{t,k}^i) \\ &\triangleq x_{t,k}^{f,i} + K_{t,k}(y_{t,k} - y_{t,k}^{f,i}). \end{aligned}$$

  (v) Sample collection: If $k > k_0$, add the sample $x_{t,k}^{a,i}$ to the set $\mathcal{X}_t$.

  end for

  end for

end for

the LEnKF is essentially immune to this problem. In the LEnKF, the importance resampling procedure draws a particle from $\mathcal{X}_{t-1}$ that matches a given particle x_t in terms of state propagation, such that the gradient $\nabla_{x_t} \log \pi(x_t | y_{1:t-1})$ can be reasonably well estimated. Then, this gradient estimate is combined with the gradient of the likelihood function of the new data y_t to update x_t. By (3.6), $\pi(x_t | y_{1:t-1})$ works as the prior distribution of x_t for the filtering distribution $\pi(x_t | y_{1:t})$. Therefore, the effect of the importance resampling procedure on the performance of the algorithm is limited if the sample size N_t is reasonably large at each stage t. In contrast, the importance resampling procedure in sequential

importance sampling draws a particle from $\mathcal{X}_{t-1}$, and treats the particle as though it were from the filtering distribution $\pi(x_t|y_{1:t})$. For high-dimensional problems, the overlap between the high-density regions of neighboring stage-filtering distributions can be very small, which naturally causes sample degeneracy. In summary, the importance resampling step of the LEnKF draws a sample for the prior distribution $\pi(x_t|y_{1:t-1})$ used at each stage t, whereas sequential importance sampling draws a sample for the target filtering distribution $\pi(x_t|y_{1:t})$. Therefore, the LEnKF is less affected by the sample degeneracy problem than is sequential importance sampling; refer to Section S1.2 for a numerical illustration.

Remark 3. (On uncertainty quantification). The LEnKF is run in an ensemble, which provides a convenient way of quantifying the uncertainty. At each stage and for each chain, the state can be estimated by averaging over iterations, as prescribed for the SGLD estimator in Teh, Thiery and Vollmer (2016) (weighted version) or Song et al. (2020) (unweighted version). The state estimates can then be further averaged over the chains. It is easy to see that the central limit theorem holds for this chain-averaged estimator approximately, given the weak dependence between different chains, and we can quantify the uncertainty accordingly.

Finally, as implied by Theorem 1, Algorithm 3 is essentially a sequential preconditioned SGLD sampler. From the proof of Theorem 2, a lower approximation error (measured in $W_2(\tilde{\pi}_t, \pi_t)$) obtained at one stage of the LEnKF helps to reduce the approximation error of the subsequent stage, and the approximation error becomes negligible as the number of stages increases.

4. Numerical Studies

4.1. Bayesian variable selection for large-scale linear regression

Consider the linear regression

$$\boldsymbol{Y} = \boldsymbol{Z}\boldsymbol{\beta} + \varepsilon, \tag{4.1}$$

where $\boldsymbol{Y} \in \mathbb{R}^N$ is the response, $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_p) \in \mathbb{R}^{N\times p}$ are covariates, $\boldsymbol{\beta} \in \mathbb{R}^p$, and $\varepsilon \sim N(0, I_N)$. An intercept term is included implicitly in the model. We generate 10 data sets from this model, with $N = 50000$, $p = 2000$, and $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p) = (1, 1, 1, 1, 1, -1, -1, -1, 0, \ldots, 0)$. That is, the first eight variables are true, and the others are false. Each variable Z_i has a marginal distribution of $N(0, I_N)$, but they are mutually correlated with a correlation coefficient of 0.5.

To conduct a Bayesian analysis for the model, we consider the following hierarchical mixture Gaussian prior distribution, which, with the latent variable $\xi_i \in \{0, 1\}$, can be expressed as

$$\begin{aligned} \beta_i|\xi_i &\sim (1-\xi_i)N(0,\tau_1^2)+\xi_i N(0,\tau_2^2), \\ P(\xi_i=1) &= 1-P(\xi_i=0)=\rho_0, \end{aligned} \tag{4.2}$$

for $i = 1, 2, \ldots, p$. Such a prior distribution is widely used in the literature on Bayesian variable selection; see, for example, George and McCullloch (1993). To apply the LEnKF to this problem, we first integrate out ξ_i from the prior (4.2), which leads to the marginal distribution

$$\beta_i \sim (1-\rho_0)N(0,\tau_1^2)+\rho_0 N(0,\tau_2^2), \quad i=1,2,\ldots,p,$$

such that the log-prior density function $\log \pi(\boldsymbol{\beta})$ is differentiable. Algorithm 2 is applied to simulate from the posterior $\pi(\boldsymbol{\beta}|\boldsymbol{Y},\boldsymbol{Z})$. In the simulation, we set $\rho_0 = 1/p = 0.0005$, $\tau_1^2 = 0.01$, and $\tau_2^2 = 1$ for the prior distribution, and set the ensemble size $m = 100$, mini-batch size $n = 100$, and learning rate $\epsilon_t = 0.2/\max\{t_0, t\}^{0.6}$, where $t_0 = 100$. As implied by Lemma S6, the convergence of the LEnKF suffers from an elbow phenomenon with respect to the mini-batch size; that is, using an extremely large mini-batch size will not lead to a much better convergence rate than using a reasonably large one. On the other hand, as discussed in Section 3.1, a large batch size can significantly increase the CPU cost of the algorithm. Therefore, a reasonably large value of n is preferred for the LEnKF. For this example, we tried $n = 100$, $n = 200$, $n = 500$, and $n = 1000$, and found that $n = 100$ leads to comparable results to those of the other three, but with a shorter CPU time. Algorithm 2 was run for 10,000 iterations, which cost 375 CPU seconds on a personal computer with a 2.9 GHz Intel Core i7 CPU and 16 GB RAM. All computations reported here were performed on the same computer.

For variable selection, we consider the factorization of the posterior distribution $\pi(\boldsymbol{\beta},\boldsymbol{\xi}|\boldsymbol{Y},\boldsymbol{Z}) \propto \pi(\boldsymbol{Y}|\boldsymbol{\beta},\boldsymbol{Z})\pi(\boldsymbol{\beta}|\boldsymbol{\xi})\pi(\boldsymbol{\xi})$, where $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_p)$. By assuming that β_i and ξ_i are *a priori* independent, we draw posterior samples of $\boldsymbol{\xi}$ from the distribution

$$\pi(\xi_{ti}=1|\beta_{ti},\boldsymbol{Y},\boldsymbol{Z}) = \frac{a_{ti}}{a_{ti}+b_{ti}}, \ i=1,2,\ldots,p, \tag{4.3}$$

where $a_{ti} = (p_0/\tau_2)\exp(-\beta_{ti}^2/2\tau_2^2)$, $b_{ti} = ((1-p_0)/\tau_1)\exp(-\beta_{ti}^2/2\tau_1^2)$, and β_{ti} denotes the posterior sample of β_i drawn by Algorithm 2 at stage t. Here, we denote by $\boldsymbol{\beta}_t = (\beta_{t1}, \beta_{t2}, \ldots, \beta_{tp})$ a posterior sample of $\boldsymbol{\beta}$ drawn by Algorithm 2 at the analysis step of stage t.

Figure 1 summarizes the variable selection results for one data set. The results for the other data sets are similar. Figure 1(a) shows the sample trajectories of $\beta_1, \beta_2, \ldots, \beta_9$, which are averaged over the ensemble and the iterations. All the samples converge weakly to their true values in 100 iterations, taking about 3.7 CPU seconds. Figure 1(b) shows the marginal inclusion probabilities of the covariates $Z_1, Z_2, \ldots, Z_p$. From this graph, each of the eight

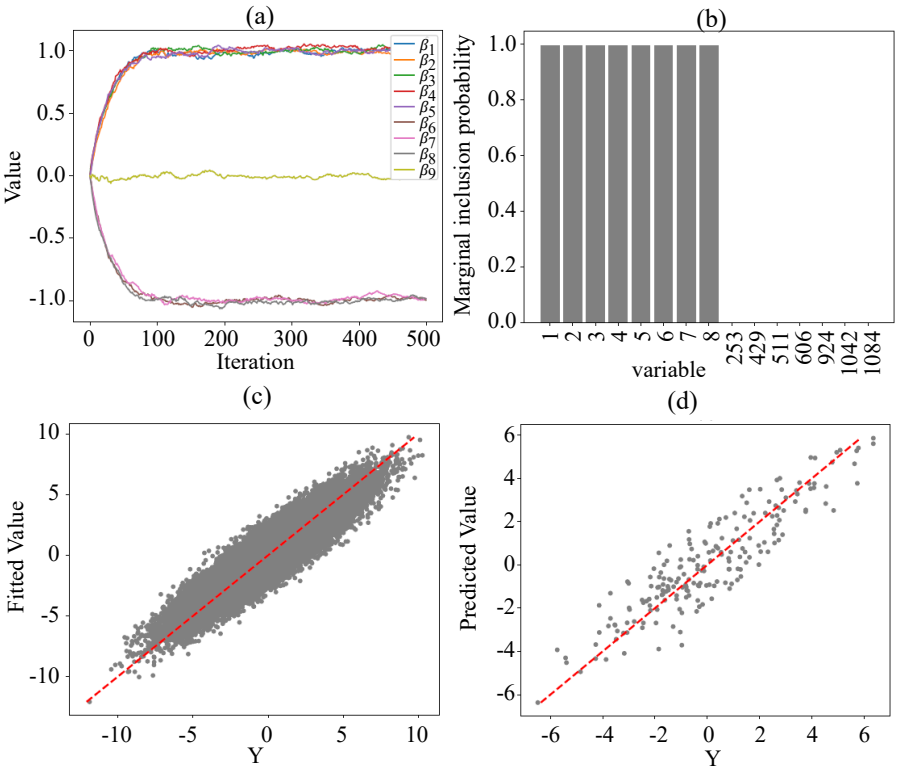

Figure 1. The LEnKF for large-scale linear regression: (a) Trajectories of $\beta_1, \ldots, \beta_9$, where $\beta_1 = \cdots = \beta_5 = 11$, $\beta_6 = \cdots = \beta_8 = -1$, and $\beta_9 = 0$ (yellow line). (b) marginal inclusion probabilities of all covariates $Z_1, \ldots, Z_p$, where the first eight high bars represent $Z_1, Z_2, \ldots, Z_8$; (c) scatter plot of Y versus the fitted value for training samples; and (d) scatter plot of Y versus the predicted value for test samples.

true variables (indexed as 1–8) has a marginal inclusion probability close to one, whereas each of the false variables has a marginal inclusion probability close to zero. Figure 1(c) shows the scatter plot of the response variable and its fitted value for the training data, and Figure 1(d) shows the scatter plot of the response variable and its predicted value for 200 test samples generated from model (4.1). In summary, Figure 1 shows that the LEnKF is able to identify true variables for a large-scale linear regression and, moreover, is extremely efficient.

For comparison, we also apply the SGLD (Welling and Teh (2011)), preconditioned SGLD (pSGLD, Li et al. (2016)), and stochastic gradient Nosé–Hoover thermostat (SGNHT, Ding et al. (2014)) to this example. For these algorithms, the learning rates are tuned to their maximum values, such that the simulation converges quickly while not exploding, and the iteration numbers are adjusted such that they cost about the same CPU time as that of the LEnKF. Refer to the Supplementary Material for their settings. Figure 2 compares the trajectories of $(\beta_1, \beta_2, \ldots, \beta_9)$ produced by the four algorithms in their first 5%

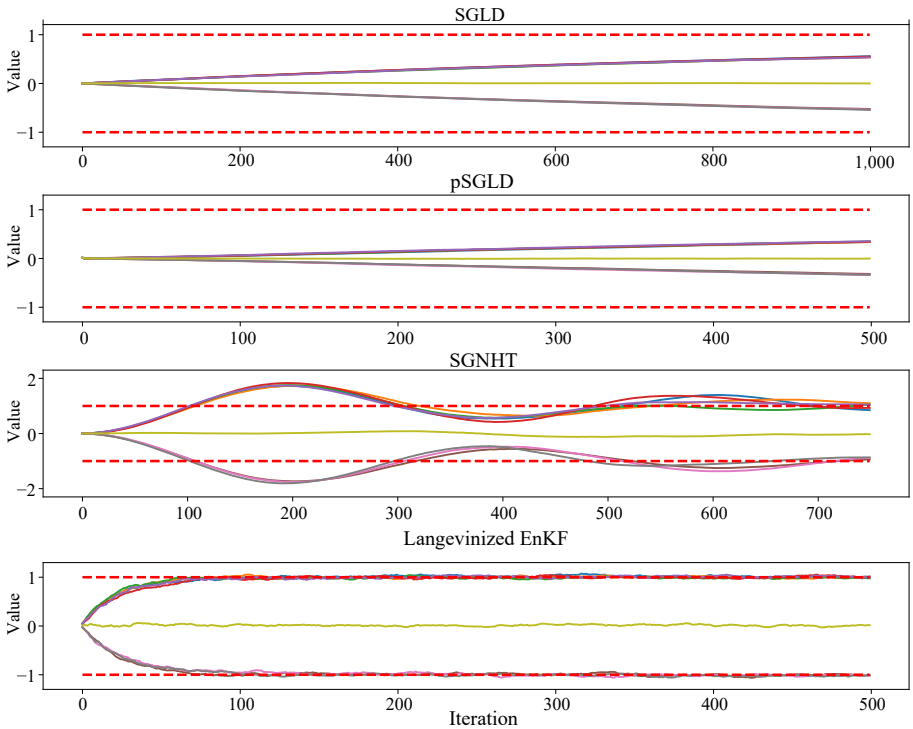

Figure 2. Trajectories of $(\beta_1, \beta_2, \ldots, \beta_9)$ produced by SGLD (upper), pSGLD (upper middle), SGNHT (lower middle), and LEnKF (lower) for a large-scale linear regression example in their first 5% iterations.

iterations, showing that the LEnKF converges significantly more quickly than SGLD, pSGLD, and SGNHT for this example, owing to the preconditioning of the LEnKF. The full trajectories of these algorithms are shown in Figure S1 of the Supplementary Material.

Because the LEnKF is a parallel preconditioned SGLD algorithm, we also compare it with parallel SGLD, pSGLD, and SGNHT. The results are presented in the Supplementary Material, showing that the LEnKF significantly outperforms parallel runs of these algorithms by a much larger margin. Recall that the LEnKF has the same computational complexity as that of the parallel SGLD, as mentioned in Section 3.1.

4.2. Uncertainty Quantification for the Lorenz-96 Model

The Lorenz-96 model was developed by Edward Lorenz in 1996 to study difficult questions related to predictability in weather forecasting (Lorenz (1996)). The model is given by

$$\frac{dx^i}{dt} = (x^{i+1} - x^{i-2})x^{i-1} - x^i + F, \quad i = 1, 2, \ldots, p,$$

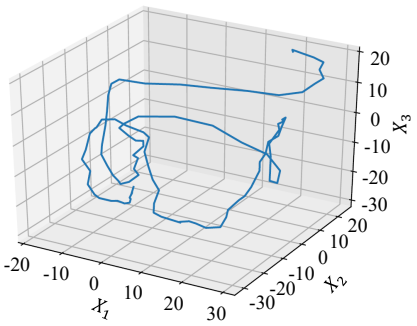

Figure 3. Chaotic path of the partial state variables (X_t^1, X_t^2, X_t^3), for $t = 1, 2, \ldots, 100$, simulated from the Lorenz-96 model.

where $F = 8$, $p = 40$, and it is assumed that $x^{-1} = x^{p-1}$, $x^0 = x^p$, and $x^{p+1} = x^1$. Here, F is known as a forcing constant, and $F = 8$ is a common value known to cause chaotic behavior. In order to generate the true state $\boldsymbol{X}_t = (X_t^1, \ldots, X_t^p)$, for $t = 1, 2, \ldots, T$, we initialize $\boldsymbol{X}_0$ by setting X_0^i to 20 for all i, but adding to X_0^{20} a small perturbation of 0.1. We solve the differential equation using the fourth-order Runge–Kutta numerical method, with a time interval of $\Delta t = 0.01$. Lastly, for each i and t, we add to X_t^i a random noise generated from $N(0, 1)$. At each stage t, data are observed for half of the state variables and masked with a Gaussian noise; that is, $y_t = H_t \boldsymbol{X}_t + \epsilon_t$, for $t = 1, 2, \ldots, T$, where $\epsilon_t \sim N(0, I_{p/2})$ and H_t is a random selection matrix. Figure 3 shows the simulated path of the partial state variables (X_t^1, X_t^2, X_t^3), for $t = 1, 2, \ldots, T$, the chaotic behavior of which indicates the challenge of the problem.

We apply Algorithm 3 to this example with the ensemble size $m = 50$, iteration number $\mathcal{K} = 20$, $k_0 = \mathcal{K}/2$, and learning rate $\epsilon_{t,k} = 0.5/k^{0.9}$, for $k = 1, 2, \ldots, \mathcal{K}$ and $t = 1, 2, \ldots, T$. At each stage t, the state is estimated by averaging over the ensembles generated at iterations $k_0 + 1, k_0 + 2, \ldots, \mathcal{K}$. The accuracy of the estimate is measured using the root mean-squared error (RMSE), defined by $RMSE_t = ||\hat{\boldsymbol{X}}_t - \boldsymbol{X}_t||_2/\sqrt{p}$, where $\hat{\boldsymbol{X}}_t$ denotes an estimate of $\boldsymbol{X}_t$. For comparison, the EnKF was applied to this example with the same ensemble size, $m = 50$. To be fair, it was run in a similar way to the LEnKF, except that we estimated the Kalman gain matrix based on the ensemble, without performing the resampling step, and drew the random error from $N(0, V_t)$ in the analysis step.

Figure 4 compares the estimates of X_t^3 produced by the LEnKF and EnKF for one simulated data set. The plots for the other components of $\boldsymbol{X}_t$ are similar. The comparison shows that the LEnKF and EnKF produce comparable RMSE_t, and that the LEnKF provides better uncertainty quantification for the estimates. Figure 4(a) shows that the confidence band by the LEnKF covers many states, but that is not the case for the EnKF. This is consistent with the existing result that the EnKF is known to provide an optimal linear estimator of the conditional

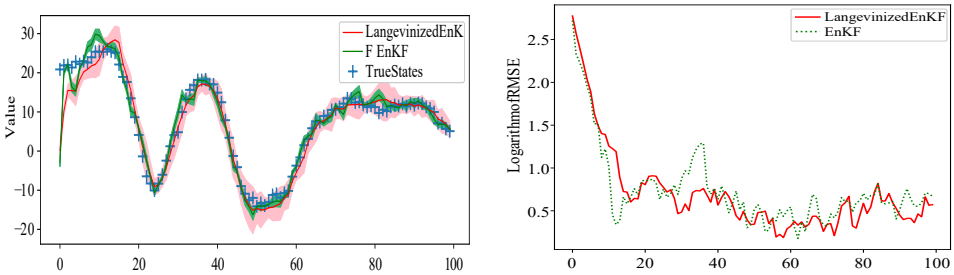

Figure 4. State estimates produced by LEnKF (red) and EnKF (green) for the Lorenz-96 model, with $t = 1, 2, \ldots, 100$: (left plot) the estimates of X_t^3, where the true states are represented by plusses, the estimates are represented by solid lines, and their 95% confidence intervals are represented by shaded bands; (right plot) $\log(\text{RMSE}_t)$ for stage t.

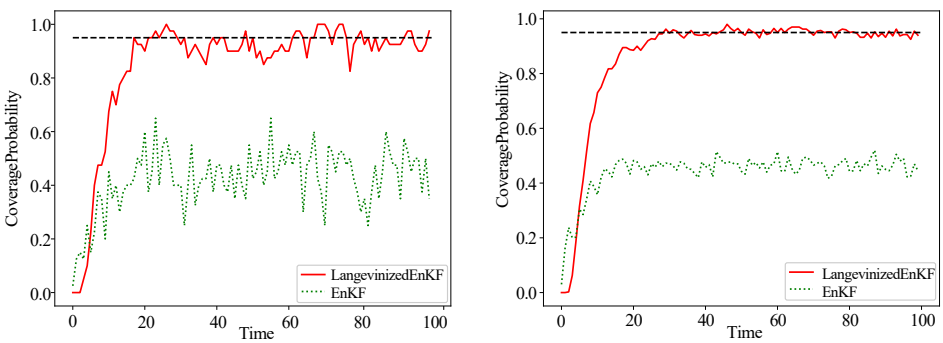

Figure 5. Coverage probabilities of the 95% confidence intervals produced by the LEnKF (solid line) and the EnKF (dotted line) for the Lorenz-96 model for stage $t = 1, 2, \ldots, 100$: (left plot) results for one data set; (right plot) results averaged over 10 data sets.

mean (Law, Tembine and Tempone (2016)), but underestimates the confidence intervals (Saetrom and Omre (2013)).

Figure 5(a) shows the coverage probabilities of the 95% confidence intervals produced by the LEnKF and EnKF, where the coverage probability is calculated by averaging over 40 state components of $\boldsymbol{X}_t$ at each stage $t \in \{1, 2, \ldots, 100\}$. Figure 5(b) shows the averaged coverage probabilities over 10 data sets. The comparison shows that the LEnKF produces a faithful coverage probability (close to its nominal level), whereas the EnKF does not. This implies that the LEnKF can correctly quantify the uncertainty of the system as t increases. This is a remarkable result, given the high nonlinearity and dynamic nature of the Lorenz-96 model.

Table 1 summarizes the results produced by the two methods on 10 data sets. For the LEnKF, we tried two choices of k_0. For each data set, we calculated the mean RMSE by averaging RMSE_t over stages $t = 21, 22, \ldots, 100$. Similarly, we calculated the mean CP by averaging CP_t over the stages $t = 21, 22, \ldots, 100$,

Table 1. Comparison of the EnKF and LEnKF, where the averages over 10 independent data sets are reported, with the standard deviation given in parentheses. The CPU time was recorded for a single run of the method.

	LEnKF		
	$k_0 = \mathcal{K}/2$	$k_0 = \mathcal{K} - 1$	EnKF
Am-RMSE	1.702(0.0343)	1.714(0.0360)	1.722(0.0230)
Am-CP	0.948(0.0028)	0.947(0.0034)	0.460(0.0029)
CPU(s)	6.37(0.3942)	3.350(0.0807)	0.817(0.0426)

where CP_t denotes the coverage probability calculated for one data set at stage t. Then, their values were further averaged over 10 data sets and denoted by "Am-RMSE" and "Am-CP," respectively. Table 1 also reports the CPU time cost of each method. Compared with the EnKF, the LEnKF produces slightly lower RMSE_t, but a much more accurate uncertainty quantification for the model. The LEnKF also produces very good results with $k_0 = \mathcal{K} - 1$, and costs much less in terms of CPU time than with $k_0 = \mathcal{K}/2$.

For the EnKF, we tried several larger ensemble sizes, up to 2000, which cost much more in terms of CPU time than the LEnKF, but with a coverage rate of only about 70%. This is consistent with the result of Law, Tembine and Tempone (2016) that the EnKF converges only to a mean-field filter, but not to the filtering distribution.

5. Extensions of the LEnKF

This section discusses a few possible extensions of the LEnKF, with numerical results reported elsewhere.

5.1. Dynamic systems with unknown parameters

Like the EnKF, the LEnKF assumes that the dynamic system contains no unknown parameters. An extension of the LEnKF to dynamic systems with unknown parameters can be achieved in several ways.

One way is to use the EM algorithm (Dempster, Laird and Rubin (1977)). Because the LEnKF is able to sample from the filtering distribution for given parameters, the EM algorithm can be conveniently used for parameter estimation. A related work is that of Aicher et al. (2019), who simulates the filtering distribution using a traditional sequential Monte Carlo algorithm. However, their method lacks the necessary scalability for big-data problems.

An alternative way is to use an adaptive stochastic gradient MCMC algorithm. Consider the model (3.2). If the propagator H_t or the observation noise covariance matrix contains unknown parameters, then the parameters can be estimated in a recursive way. In this case, the following step can be added to Algorithm 2:

(iv) **Parameter updating:** Update the parameters by recursion $\boldsymbol{\vartheta}_t = (1 - a_t)\boldsymbol{\vartheta}_{t-1} + a_t\phi(\boldsymbol{\vartheta}_{t-1}, \boldsymbol{x}_t^a)$, where $\boldsymbol{\vartheta}$ denotes the vector of unknown parameters, $\{a_t\}$ is a prespecified, positive decreasing sequence satisfying the conditions $\sum_t a_t = \infty$ and $\sum_t a_t^2 < \infty$, $\boldsymbol{x}_t^a = (x_t^{a_1}, \ldots, x_t^{a_m})$ denotes the ensemble of samples at stage t, and $\phi(\boldsymbol{\vartheta}_{t-1}, \boldsymbol{x}_t^a)$ is a mapping to an estimate of $\boldsymbol{\vartheta}$ based on the ensemble $\boldsymbol{x}_t^a$.

Using the theory of stochastic approximation (Robbins and Monro (1951)), the mapping $\phi(\boldsymbol{\vartheta}_{t-1}, \boldsymbol{x}_t^a)$ can be easily designed. With the parameter updating step, the LEnKF becomes an adaptive stochastic gradient MCMC algorithm, where the target distribution varies between iterations. The convergence of such an adaptive algorithm can be studied in a similar way to Deng et al. (2019). Under appropriate conditions, we can show that as $t \to \infty$, $\boldsymbol{\vartheta}_t$ converges to the true parameters in probability, and $\boldsymbol{x}_t^a$ converges weakly to the filtering distribution.

Finally, note that we can also use the state-augmentation approach for the parameter estimation. However, this approach applies only when the resulting covariance matrix $\Sigma_t = (n/N)(I - K_t H_t)$ is still a constant matrix of the augmented state variable. Otherwise, the weak convergence of $\boldsymbol{x}_t^a$ to the filtering distribution is no longer guaranteed.

5.2. Dynamic systems with non-Gaussian observations

In practice, we often encounter problems in which the response variable follows a non-Gaussian distribution, such as a multinomial or Poisson distribution. The LEnKF can be extended to these problems by introducing a latent variable. For example, consider an inverse problem for which the latent variable model can be formulated as

$$z|y \sim \rho(z|y), \quad y = h(x) + \eta, \quad \eta \sim N(0, \Gamma), \tag{5.1}$$

where z is the observed data following a non-Gaussian distribution $\rho(\cdot)$, y is the latent Gaussian variable, and x is a parameter. To adapt the LEnKF to simulate from the posterior distribution $\pi(x|z)$, we need only add an imputation step to Algorithm 2, between the forecast and the analysis steps. The imputation step simulates a latent vector y from the distribution $\pi(y|x, z) \propto \rho(z|y)f(y|x)$. Because the imputation leads to an unbiased estimate for the gradient of the involved log-density function, the proposed extension is valid, and we can use it to generate samples from the target posterior $\pi(x|z)$. A further extension of this algorithm to data assimilation problems is straightforward.

5.3. Dynamic systems with nonlinear measurement equations

As indicated by Algorithm 3, the LEnKF is a sequential preconditioned SGLD sampler. At each stage, it aims to simulate from the posterior distribution for a linear inverse problem using an appropriately designed prior distribution,

for which the gradient of the log-density function is estimated based on the samples simulated in the preceding stage. In the same vein, Algorithm 3 can be extended to data-assimilation problems with nonlinear measurement equations, for which we need only determine how the LEnKF can be used for nonlinear inverse problems.

Consider the nonlinear inverse problem

$$y = \mathcal{G}(z) + \eta, \quad \eta \in N(0, \Gamma),$$

where $y = (\tilde{y}_1^T, \tilde{y}_2^T, \ldots, \tilde{y}_B^T)^T$, $\Gamma = \text{diag}[V, V, \ldots, V]$ is a diagonal block matrix, each block V is of size $n \times n$, and $N = Bn$, for some positive constant B. To reformulate the problem in the central dynamic system (1.1), we define an augmented state vector by an n-vector γ_t:

$$x_t = (z^T, \gamma_t^T)^T, \quad \gamma_t = \mathcal{G}_t(z) + u_t, \quad u_t \sim N(0, \alpha V), \tag{5.2}$$

where $\mathcal{G}_t(\cdot)$ is the mean response function for a mini-batch of data drawn at stage t, and $0 < \alpha < 1$ is a prespecified constant. In this paper, α is called the variance-splitting proportion.

Let $\pi(z)$ denote the prior density function of z, which is differentiable with respect to z. The conditional distribution of γ_t is $\gamma_t | z \sim N(\mathcal{G}_t(z), \alpha V)$, and the joint density function of x_t is then $\pi(x_t) = \pi(z)\pi(\gamma_t|z)$. Based on Langevin dynamics, a system identical to (3.2) (in symbol) can be constructed for the nonlinear inverse problem:

$$\begin{aligned} x_t &= x_{t-1} + \epsilon_t \frac{n}{2N} \nabla_x \log \pi(x_{t-1}) + w_t \\ y_t &= H_t x_t + v_t, \end{aligned} \tag{5.3}$$

where $w_t \sim N(0, (n/N)Q_t)$, $Q_t = \epsilon_t I_p$, and p is the dimension of x_t; $H_t = (0, I)$, such that $H_t x_t = \gamma_t$; $v_t \sim N(0, (1-\alpha)V)$, which is independent of w_t for all t; and y_t is a random draw from $\{\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_B\}$.

Using this formulation, the LEnKF can be easily extended to simulate from the posterior $\pi(z|y)$ for the nonlinear inverse problem. Using the variance-splitting state-augmentation approach, in the same vein as Algorithm 3, the LEnKF can be further extended to data-assimilation problems with nonlinear measurement equations.

6. Conclusion

We have proposed the LEnKF as a scalable particle filter by reformulating the EnKF under the framework of Langevin dynamics. The LEnKF is a sequential preconditioned SGLD algorithm, but its execution is accelerated using a forecast-analysis procedure. The LEnKF converges to the correct filtering distribution

in terms of the 2-Wasserstein distance as the number of iterations per stage increases. We can apply the LEnKF to state estimation for both inverse and data-assimilation problems, and quantify the uncertainty of the states. The LEnKF is not only scalable with respect to the state dimension and sample size, but also tends to be immune to the sample degeneracy problem encountered by conventional particle filters.

Supplementary Material

The online Supplementary Material presents the proofs of Theorem 1 and Theorem 2, as well as additional numerical examples.

Acknowledgments

This research was supported in part by the NSF grants DMS-1811812 (Song) and DMS-2015498 (Liang) and NIH grant R01-GM126089 (Liang). The authors thank the reviewers for their insightful and helpful comments.

References

Aanonsen, S., Naevdal, G., Oliver, D., Reynolds, A. and Valles, B. (2009). The ensemble Kalman filter in reservoir engineering–A review. *SPE Journal* **14**, 393–412.

Aicher, C., Putcha, S., Nemeth, C., Fearhead, P. and Fox, E. (2019). Stochastic gradient MCMC for nonlinear state space models. *SIAM J. MATH. Data SCI.* **1**, 557–587.

Anderson, J. (2001). An ensemble adjustment filter for data assimilation. *Monthly Weather Review* **129**, 2884–2903.

Baek, S.-J., Hunt, B., Kalney, E., Ott, E. and Szunyogh, I. (2006). Local ensemble Kalman filtering in the presence of model bias. *Tellus* **58A**, 293–306.

Bardenet, R., Doucet, A. and Holmes, C. C. (2017). On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research* **18**,1–43.

Bergou, E. H., Gratton, S. and Mandel, J. (2019). On the convergence of a non-linear ensemble Kalman smoother. *Applied Numerical Mathematics* **137**, 151–168.

Beskos, A., Jasra, A., Kantas, N. and Thiery, A. (2016). On the convergence of adaptive sequential Monte Carlo methods. *The Annals of Applied Probability* **26**, 1111–1146.

Bierkens, J., Fearnhead, P. and Roberts, G. (2019). The Zig-Zag process and super-efficient Monte Carlo for Bayesian analysis of big data. *The Annals of Statistics* **47**, 1288–1320.

Bouchard-Côté, A., Vollmer, S. J. and Doucet, A. (2018). The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method. *Journal of the American Statistical Association* **113**, 855–867.

Cappé, O., Guillin, A., Marin, J. M. and Robert, C. P. (2004). Population Monte Carlo. *Journal of Computational and Graphical Statistics* **13**, 907–929.

Chen, H., Seita, D., Pan, X. and Canny, J. (2018). An efficient minibatch acceptance test for Metropolis-Hastings. In *IJCAI*, 5359–5363.

Chen, T., Fox, E. B. and Guestrin C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *ICML*, 1683–1691.

Crisan, D. and Doucet, A. (2000). Convergence of sequential Monte Carlo methods. Technical report. Department of Engineering, University of Cambridge.

Dempster, A., Laird, N. and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B (Methodological)* **39**, 1–38.

Deng, W., Zhang, X., Liang, F. and Lin, G. (2019). An adaptive empirical Bayesian method for sparse deep learning. In *NeurIPS*, 5563–5573.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D. and Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. In *NIPS*.

Doucet, A., de Freitas, N. and Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer, New York.

Ernst, O., Sprungk, B. and Starkloff, H.-J. (2015). Analysis of the ensemble and polynomial chaos Kalman filters in Bayesian inverse problems. *SIAM/ASA J. Uncertainty Quantification* **3**, 823–851.

Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans* **99**, 10143–10162.

Evensen, G. and van Leeuween, P. J. (1996). Assimilation of geosat altimeter data for the Agulhas current using the ensemble Kalman filter with a quasi-geostrophic model. *Monthly Weather Review* **124**, 85–96.

George, E. and McCulloch, R. (1993). Variable selection via Gibbs sampling. *Journal of the American Statistical Association* **88**, 881–889.

Gillijns, S. and De Moor, B. (2007). Model error estimation in ensemble data assimilation. *Nonlinear Processes in Geophysics* **14**, 59–71.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation* **9**, 1735–1780.

Houtekamer, P. and Mitchell, H. (2011). A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review* **129**, 123–137.

Iglesias, M., Law, K. and Stuart, A. (2013). Ensemble Kalman methods for inverse problems. *Inverse Problems* **29**, 045001.

Kwiatkowski, E. and Mandel, J. (2015). Convergence of the square root ensemble Kalman filter in the large ensemble limit. *SIAM/ASA J. Uncertainty Quantification* **3**, 1–17.

Law, K., Tembine, H. and Tempone, R. (2016). Deterministic mean-field ensemble Kalman filtering. *SIAM J. Sci. Comput.* **38**, A1251–A1279.

Le Gland, F., Monbet, V. and Tran, V.-D. (2009). Large sample asymptotics for the ensemble Kalman filter. Technical Report. INRIA RR-7014, inria-00409060.

Li, C., Chen, C., Carlson, D. and Carin, L. (2016). Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 1788–1794. AAAI Press.

Li, C., Srivastava, S. and Dunson, D. (2017). Simple, scalable and accurate posterior interval estimation. *Biometrika* **104**, 665–680.

Liu, J. S. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association* **93**, 1032–1044.

Lorenz, E. (1996). Predictability–a problem partly solved. In *Predictability of Weather and Climate* (Edited by T. Palmer and R. Hagedorn), 40–58. Cambridge University Press.

Ma, Y.-A., Chen, T. and Fox, E. B. (2015). A complete recipe for stochastic gradient MCMC. In *NIPS*, 2917–2925.

Maclaurin, D. and Adams, R. P. (2014). Firefly Monte Carlo: Exact MCMC with subsets of data. In *IJCAI*, 4289–4295.

Nemeth, C. and Fearnhead, P. (2021). Stochastic gradient Markov chain Monte Carlo. *Journal of the American Statistical Association* **116**, 433–450.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics* **22**, 400–407.

Saetrom, J. and Omre, H. (2013). Uncertainty quantification in the ensemble Kalman filter. *Scandinavian Journal of Statistics* **40**, 868–885.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I. and McCulloch, R. E. (2016). Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management* **11**, 78–88.

Shumway, R. and Stoffer, D. (2006). *Time Series Analysis and Its Applications with R Examples*. Springer, New York.

Song, Q., Sun, Y., Ye, M. and Liang, F. (2020). Extended stochastic gradient MCMC algorithms for large-scale Bayesian variable selection. *Biometrika* **107**, 997–1004.

Srivastava, S., Li, C. and Dunson, D. B. (2018). Scalable Bayes via Barycenter in Wasserstein space. *Journal of Machine Learning Research* **19**, 1–35.

Teh, W., Thiery, A. and Vollmer, S. (2016). Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research* **17**, 1–33.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, 681–688.

Peiyi Zhang

Department of Statistics, Purdue University, West Lafayette, IN 47906, USA.

E-mail: z.peiyi1993@gmail.com

Qifan Song

Department of Statistics, Purdue University, West Lafayette, IN 47907, USA.

E-mail: qfsong@purdue.edu

Faming Liang

Department of Statistics, Purdue University, West Lafayette, IN 47907, USA.

E-mail: fmliang@purdue.edu

(Received May 2022; accepted December 2022)