THE UNIVERSITY OF CHICAGO

DATA-DRIVEN METHODS FOR INVERSE PROBLEMS: BLENDING DATA ASSIMILATION AND MACHINE LEARNING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

COMMITTEE ON COMPUTATIONAL AND APPLIED MATHEMATICS

BY
YUMING CHEN

CHICAGO, ILLINOIS
JUNE 2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Daniel Sanz-Alonso. He has been a constant source of inspiration, providing patient and thoughtful feedback on my work, and always being responsive to my questions and concerns. His expertise and dedication to excellence have been instrumental in shaping my research direction and helping me grow as a scholar. My Ph.D. journey would be impossible without his unwavering support.

I would also like to express my sincere gratitude to Rebecca Willett, for giving me the opportunity to join her research group and learn about cutting-edge machine learning techniques that complement my background. I am deeply grateful for her valuable feedback on my work. I would also like to thank Mihai Anitescu for serving on my thesis committee and providing insightful comments on my research.

I would like to extend my thanks to all the faculty and staff in the Committee on Computational and Applied Mathematics. Thanks to Mary Silber, Zellencia Harris and Jonathan Rodriguez, for their administrative support and continuous care throughout my time in the program.

I would also like to express my appreciation to my peers, including Yian Chen, Zhen Dai, Ruiyi Yang, Ruoxi Jiang, Qin Wen, Haotian Gu, Hongpeng Guo, and many others. I would like to thank them for the good research discussions we've had, as well as happiness and joy they've brought into my five-year Ph.D. journey.

Last but not least, I would like to thank my parents for their unconditional support. Their encouragement has been essential for me to pursue this career path. I would also like to extend my heartfelt thanks to Wenjun Wang, whose love and support make this Ph.D. journey even more fulfilling and memorable.

ABSTRACT

Inverse problems (IPs) deal with the task of reconstructing input variables from noisy observations, defined through a forward model. When the forward map is known, the reconstruction of input variables can be performed via numerous approaches, including optimization-based algorithms (e.g., maximum likelihood estimation) and sampling-based algorithms (e.g., Markov chain Monte Carlo). However, there are scenarios where: (1) we do not have perfect knowledge about the forward model; or (2) the forward model is expensive to simulate, which severely limits the implementation of optimization and sampling algorithms that may require multiple forward model simulations. Therefore, in these scenarios, it is essential to approximate the forward model by a parameterized surrogate model. We propose data-driven approaches that jointly learn the parameters of the surrogate model, and reconstruct input variables, from observation data alone.

Data assimilation (DA) deals with the task of reconstructing temporally evolving hidden states from noisy time series observations, defined through a state space model (SSM). When the SSM is known, the reconstruction of states can be performed using optimization-based algorithms (e.g., 4DVAR) or sampling-based algorithms (e.g., particle filtering). However, similar to IPs, there are scenarios where: (1) we do not have perfect knowledge about the SSM; or (2) the SSM is expensive to simulate, which increases the computational cost of the reconstruction algorithms. Therefore, in these scenarios, it is essential to approximate the SSM with a parameterized surrogate model. We employ machine learning techniques and propose data-driven approaches that jointly learn the parameters of the surrogate model, and reconstruct the states, from observation data alone.

CHAPTER 1

INTRODUCTION

1.1 What are Inverse Problems?

Inverse problems (IPs) deal with the task of reconstructing input $u \in \mathbb{R}^{d_u}$ from noisy observation $y \in \mathbb{R}^{d_y}$, defined through a forward model:

$$y = \mathcal{A}(u) + \eta, \qquad u \sim p_u(u), \tag{1.1.1}$$

where $\eta \sim \mathcal{N}(0, \Gamma)$ represents Gaussian measurement error, and p_u represents the *prior* distribution of u. The random variables η and u are assumed to be independent. The solution to this inverse problem can be described in terms of the *posterior* distribution:

$$\pi(u) := p(u|y) = \frac{1}{Z} p_u(u) \mathcal{N}\big(y; \mathcal{A}(u), \Gamma\big) \tag{1.1.2}$$

where Z is a normalizing constant that does not depend on u. Throughout this thesis, we use the notation $\mathcal{N}(x; m, C)$ to denote the Gaussian density with mean m and covariance C evaluated at x, and the corresponding Gaussian distribution is denoted by $\mathcal{N}(m, C)$. This chapter introduces different algorithms to approximate the posterior distribution (1.1.2), under the assumption that $\mathcal{A}$ is known. They can be broadly classified into two categories: optimization-based algorithms and sampling-based algorithms.

1.1.1 Optimization-based Algorithms for Inverse Problems

In applications, since u is often high-dimensional, it is nontrivial to visualize the posterior distribution (1.1.2). Instead, it is useful to summarize the posterior distribution with a few numerical values. In this section, we focus on the posterior mode or *maximum a posteriori (MAP)* estimator, which corresponds to the maximizer of Eq. (1.1.2). Equivalently, it is defined as the solution to the following optimization problem:

$$u_{\text{MAP}} := \arg\min_u \mathsf{J}(u), \qquad \text{where } \mathsf{J}(u) := \frac{1}{2}\|y - \mathcal{A}(u)\|_\Gamma^2 - \log p_u(u) \tag{1.1.3}$$

We use the notation $\|v\|_\Gamma = \sqrt{v^\top \Gamma^{-1} v}$ throughout the thesis. Common choices for the prior distribution p_u include the Gaussian distribution, which turns (1.1.3) into an L_2-regularized optimization problem, and the Laplace distribution, which turns (1.1.3) into an L_1-regularized optimization problem.

We will highlight three optimization algorithms for (1.1.3) that will be useful for later chapters: the gradient descent method, the Gauss-Newton method and the Levenberg-Marquardt method. The first two algorithms can be regarded as line search methods, while the last one can be regarded as a trust region method. Both Gauss-Newton and Levenberg-Marquardt are only applicable to the cases where p_u is Gaussian. We refer the reader to [Nocedal and Wright, 1999] for a general introduction to optimization algorithms and their theoretical properties.

Gradient Descent Method The gradient descent method is commonly used in machine learning. Assume that the gradient of the objective function $\nabla\mathsf{J}(u)$ is available, the algorithm iteratively adjusts u by taking small steps in the opposite direction of $\nabla\mathsf{J}(u)$:

$$u_{k+1} = u_k + \alpha_k s_k, \qquad k = 0, 1, \ldots, \tag{1.1.4}$$

with $s_k = -\nabla\mathsf{J}(u_k)$ and appropriate choices of step size α_k. With modern development of machine learning computational software, accurate description of the gradient $\nabla\mathsf{J}(u)$ becomes available for a wide range of applications, through automatic differentiation libraries like PyTorch [Paszke et al., 2019], JAX [Bradbury et al., 2018], and Tensorflow [Abadi et al., 2016]. One can also compute $\nabla\mathsf{J}$ when $\mathcal{A}$ involves ordinary or stochastic differential equation solvers [Chen et al., 2018; Li et al., 2020a].

The gradient descent method has limitations, perhaps most notably that the algorithm may get stuck in local minima. To alleviate this issue and accelerate its convergence, several advanced techniques have been developed, e.g. momentum, adaptive step sizes, etc. Moreover, despite the aforementioned recent advances in software, computation of $\nabla \mathsf{J}(u)$ can still be expensive or impossible, e.g. for certain problems that involve partial differential equations (PDEs). Despite these caveats, gradient descent is a powerful method for finding the MAP estimator, as well as minimizers to more general optimization objectives J. We will explore this method with automatic differentiation techniques in Chapter 3 and Chapter 4.

Gauss-Newton Method When the prior distribution p_u is Gaussian, the optimization problem Eq. (1.1.3) can be transformed into a weighted least-squares problem. To see this, assume $u \sim \mathcal{N}(m, C)$, and we have

$$\begin{aligned} \mathsf{J}(u) &= \frac{1}{2}\|y - \mathcal{A}(u)\|_\Gamma^2 + \frac{1}{2}\|u - m\|_C^2 \\ &= \frac{1}{2}\|z - g(u)\|_S^2, \end{aligned} \tag{1.1.5}$$

where

$$z := \begin{bmatrix} y \\ m \end{bmatrix}, \qquad g(u) := \begin{bmatrix} \mathcal{A}(u) \\ u \end{bmatrix}, \qquad S := \begin{bmatrix} \Gamma & 0 \\ 0 & C \end{bmatrix}.$$

Two common ways to find the minimizer of the objective in the form of Eq. (1.1.5) are the Gauss-Newton method and the Levenberg-Marquardt method. Gauss-Newton is a line search method that has the form of Eq. (1.1.4) and uses second-order derivative information of J. However, instead of computing the Newton direction $s_k^{\mathrm{N}} = -\big(\nabla^2 \mathsf{J}(u_k)\big)^{-1} \nabla \mathsf{J}(u_k)$ directly, we approximate the Hessian matrix $\nabla^2 \mathsf{J}(u_k)$ using first-order derivative information of g, ignoring the second-order terms that are hard to deal with:

$$\nabla^2 \mathsf{J}(u_k) \approx \mathrm{D}g(u_k)^\top S^{-1} \mathrm{D}g(u_k) \tag{1.1.6}$$

where $Dg(u_k)$ is the Jacobian matrix of g at u_k, and the approximation follows from Eq. (1.1.5). Therefore, the Gauss-Newton method is defined using the same form as Eq. (1.1.4), but with $s_k = -\big(\mathrm{D}g(u_k)^\top S^{-1} \mathrm{D}g(u_k)\big)^{-1} \nabla \mathsf{J}(u_k)$ and appropriate choices of step size α_k.

Levenberg-Marquardt Method The Levenberg-Marquardt method uses a trust-region approach, where at each iteration k, we find the solution s_k to the sub-problem:

$$\arg\min_s m_k(s) := \mathsf{J}(u_k) + \nabla \mathsf{J}(u_k)^\top s + \frac{1}{2} s^\top B_k s \qquad \text{s.t. } \|s\| \le \delta_k \tag{1.1.7}$$

and set $u_{k+1} = u_k + s_k$. Ideally, B_k is a positive definite matrix that is close to the Hessian $\nabla^2 \mathsf{J}(u_k)$, so that $m_k(s)$ is a good quadratic approximation of $\mathsf{J}(u_k + s)$ when $\|s\|$ is small. The value of δ_k is tuned so that $m_k(s)$ does not deviate too much from $\mathsf{J}(u_k + s)$.

For the least-squares problem Eq. (1.1.5), it is thus natural to reuse the Hessian approximation Eq. (1.1.6) and set

$$B_k = \mathrm{D}g(u_k)^\top S^{-1} \mathrm{D}g(u_k) \tag{1.1.8}$$

in Eq. (1.1.7), avoiding computation of second-order derivatives of g. Then the optimization problem Eq. (1.1.7) can be solved through analysis of the KKT conditions.

We will come back to the Gauss-Newton method and the Levenberg-Marquardt method in Chapter 5, where we introduce new ways to approximate the Jacobian term $\mathrm{D}g(u_k)$ through particles, and study theoretical properties of these algorithms in continuous time limit on certain classes of inverse problems.

1.1.2 Sampling-based Algorithms for Inverse Problems

Another way to summarize the posterior distribution (1.1.2) is to deliver approximate samples from it. More specifically, we aim to approximate the following integral for any test function h:

$$I_\pi[h] := \int h(u)\pi(u)\mathrm{d}u \tag{1.1.9}$$

with approximate samples. In this section, we review importance sampling and Markov chain Monte Carlo (MCMC), two classes of sampling algorithms that generate approximate samples from a target distribution π.

Importance Sampling To provide samples from the target distribution $\pi(u)$, we can first sample $\{u^1, \ldots, u^N\}$ from a proposal distribution $q(u)$, and assign the n-th sample with weight $w^n = \pi(u^n)/q(u^n)$. The weighted particles $\{(w^1, u^1), \ldots, (w^N, u^N)\}$ can be regarded as approximate samples from the target distribution, which gives the following approximation:

$$\mathsf{I}_\pi^{\mathrm{IS}}[h] := \frac{1}{N}\sum_{n=1}^{N} w^n h(u^n) \approx \mathsf{I}_\pi(h). \tag{1.1.10}$$

It is often the case that $\pi(u)$ is only known up to a normalizing constant, making it impossible to compute w^n in Eq. (1.1.10). For example, this is the case for Eq. (1.1.2) when $\mathcal{A}$ is nonlinear or p_u is non-Gaussian, as Z is hard to compute. Let $\widetilde{\pi}(u) := Z\pi(u)$ where Z is some normalizing constant and $\widetilde{\pi}$ is easily accessible. We can compute the self-normalizing weights:

$$w^n = \frac{\widetilde{w}^n}{\sum_{i=1}^N \widetilde{w}^i}, \qquad \text{where } \widetilde{w}^n = \frac{\widetilde{\pi}(u^n)}{q(u^n)}, \tag{1.1.11}$$

which leads to the following approximation:

$$\mathsf{I}_\pi^{\mathrm{AIS}}[h] := \sum_{n=1}^{N} w^n h(u^n). \tag{1.1.12}$$

The idea behind Eq. (1.1.11) is that $\frac{1}{N}\sum_{i=1}^{N}\widetilde{w}^i$ is an unbiased estimator of Z:

$$\mathbb{E}_{u^1,\dots,u^N\sim q(u)}\left[\frac{1}{N}\sum_{i=1}^{N}\widetilde{w}^i\right]=\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{u^i\sim q(u)}\left[\frac{\widetilde{\pi}(u^i)}{q(u^i)}\right]=\frac{1}{N}NZ=Z. \tag{1.1.13}$$

For the posterior distribution in the form of Eq. (1.1.2), it is useful to set the proposal distribution to be the prior, i.e., $q=p_u$. Then the unnormalized weights $\widetilde{w}^n$ in Eq. (1.1.11) corresponds to the data likelihood $p(y|u^n)$, and Z corresponds to the marginal likelihood $p(y)$, also known as *model evidence.* Importance sampling is a fundamental building block of several data assimilation algorithms, and will be discussed in detail in later chapters. Moreover, the estimation of model evidence Z is a key component of Chapter 3 and Chapter 4.

Metropolis-Hastings Algorithm The Metropolis-Hastings algorithm describes a general MCMC methodology that generates a Markov kernel $\mathbb{K}(u^{n+1}|u^n)$ with the target distribution $\pi(u)$ as its invariant distribution. We first define a kernel $q(v|u)$ that is easy to sample from, which we call the *proposal.* At each iteration n, we propose $v\sim q(v|u^n)$ as a candidate for u^{n+1}, accept it with probability $a(u^n,v)$, defined by

$$a(u,v)=\min\left\{1,\frac{\pi(v)q(u|v)}{\pi(u)q(v|u)}\right\}, \tag{1.1.14}$$

and otherwise set $u^{n+1}=u^n$. The resulting Markov kernel $\mathbb{K}(v|u)=a(u,v)q(v|u)+\big(1-a(u,v)\big)\delta_u(v)$ satisfies the detailed balance equation w.r.t. the target distribution $\pi(u)$:

$$\pi(u)\mathbb{K}(v|u)=\pi(v)\mathbb{K}(u|v), \tag{1.1.15}$$

and therefore has $\pi(u)$ as the invariant distribution. The samples $\{u^1,\dots,u^n,\dots\}$ can be treated as approximate samples from $\pi(u)$. In other words, for any test function h, we have the following approximation:

$$\mathsf{I}_\pi^{\mathrm{MH}}[h] := \frac{1}{N}\sum_{n=1}^{N} h(u^n) \approx \mathsf{I}_\pi(h). \tag{1.1.16}$$

Common choices of the proposal kernel $q(v|u)$ include uniform distribution over some interval $[u-\gamma, u+\gamma]$, and Gaussian distribution centered at u. In both cases q is easy to sample from, symmetric around u, and satisfies $q(u|v) = q(v|u)$ so that Eq. (1.1.14) is easy to compute.

Preconditioned Crank-Nicolson (pCN) Algorithm The pCN algorithm is a special type of MH algorithm with a specific choice of proposal $q(v|u)$. It is suited for cases where $\pi(u)$ has the form of a posterior distribution Eq. (1.1.2) and is high-dimensional. Expanding Eq. (1.1.14), we have:

$$a(u,v) = \min\left\{1, \frac{p_u(v)\mathcal{N}\big(y;\mathcal{A}(v),\Gamma\big)q(u|v)}{p_u(u)\mathcal{N}\big(y;\mathcal{A}(u),\Gamma\big)q(v|u)}\right\}. \tag{1.1.17}$$

An issue of the symmetric proposal kernels discussed in the previous paragraph is that, although $q(u|v) = q(v|u)$, the ratio $p_u(v)/p_u(u)$ can be very small when u is high-dimensional. This is especially the case when, e.g., p_u is a Gaussian distribution $\mathcal{N}(m, C)$ where C has rapidly decaying eigenvalues. This leads to an acceptance probability that is very close to 0, making the Markov chain inefficient.

The pCN algorithm proposes a kernel $q(v|u)$ that satisfies $p_u(v)q(u|v) = p_u(u)q(v|u)$, i.e., detailed balance equation w.r.t. the prior p_u. When p_u is a Gaussian distribution $\mathcal{N}(m, C)$, it is easy to check that

$$q(v|u) := \mathcal{N}\big(v; m + \sqrt{1-\beta^2}(u-m), \beta^2 C\big) \tag{1.1.18}$$

satisfies the desired property for any choice of $\beta \in (0, 1)$. Therefore, Eq. (1.1.17) is simplified to $a(u,v) = \min\left\{1, \frac{\mathcal{N}\left(y;\mathcal{A}(v),\Gamma\right)}{\mathcal{N}\left(y;\mathcal{A}(u),\Gamma\right)}\right\}$. We will investigate the application of pCN to a family of high-dimensional inverse problems in Chapter 2.

1.2 What is Data Assimilation?

Data assimilation (DA) deals with the task of reconstructing temporally evolving hidden states $u_t \in \mathbb{R}^{d_u}$ from noisy time series observations $y_t \in \mathbb{R}^{d_y}$, defined through the following state space model (SSM):

$$u_t = F(u_{t-1}) + \xi_t, \qquad \xi_t \sim \mathcal{N}(0,Q), \qquad 1 \le t \le T, \tag{1.2.1}$$

$$y_t = H_t u_t + \eta_t, \qquad \eta_t \sim \mathcal{N}(0,R_t), \qquad 1 \le t \le T, \tag{1.2.2}$$

$$u_0 \sim p_u(u_0), \tag{1.2.3}$$

where ξ_t represents Gaussian model error, η_t represents Gaussian measurement error and p_u represents the prior distribution on u_0. The random variables $\{\xi_t\}_{t=1}^T$, $\{\eta_t\}_{t=1}^T$ and u_0 are all assumed to be independent. For simplicity, we write $u_{t_0:t_1} := \{u_t\}_{t=t_0}^{t_1}$.

The solution to the data assimilation task can be described in terms of the *filtering* distribution $p(u_t|y_{1:t})$ and the *smoothing* distribution $p(u_t|y_{1:T})$, for $1 \le t \le T$. The difference between the two types of distributions is whether the state u_t is inferred based on observations up to time t or across the entire timespan. This chapter introduces different approaches to sequentially approximate the filtering distribution $p(u_t|y_{1:t})$, as it aligns with the materials discussed in later chapters. For this chapter, we assume that the state space model (1.2.1)-(1.2.3) is known perfectly (i.e., F, Q, H_t, R_t, p_u are known).

1.2.1 Data Assimilation as Sequential Inverse Problems

In this section, we introduce the general principle of deriving the filtering distribution: iteratively updating from $p(u_{t-1}|y_{1:t-1})$ to $p(u_t|y_{1:t})$. Assume that $p(u_{t-1}|y_{1:t-1})$ is given, the process can be decomposed into two steps, known as the *prediction* step and the *analysis* step:

$$\text{(prediction)} \qquad p(u_t|y_{1:t-1}) = \int p(u_{t-1}|y_{1:t-1})\mathcal{N}\big(u_t; F(u_{t-1}), Q\big)\, \mathrm{d}u_{t-1}, \tag{1.2.4}$$

$$\text{(analysis)} \qquad p(u_t|y_{1:t}) = \frac{1}{Z_t} p(u_t|y_{1:t-1})\mathcal{N}(y_t; H_t u_t, R_t). \tag{1.2.5}$$

The *predictive* distribution $p(u_t|y_{1:t-1})$ can be interpreted as the distribution of u_t after applying the transformation (1.2.1) to $u_{t-1} \sim p(u_{t-1}|y_{1:t-1})$. The filtering distribution $p(u_t|y_{1:t})$ can be interpreted as the solution to the following inverse problem for u_t, which is a special case of (1.1.1):

$$y_t = H_t u_t + \eta_t, \qquad u_t \sim p(u_t|y_{1:t-1}), \tag{1.2.6}$$

where $\eta_t \sim \mathcal{N}(0, R_t)$. Here the predictive distribution $p(u_t|y_{1:t-1})$ is treated as the prior, and the filtering distribution $p(u_t|y_{1:t})$ corresponds to the posterior distribution of this inverse problem with a linear forward map.

1.2.2 Filtering Algorithms for Data Assimilation

Depending on how the inverse problem (1.2.6) is approached, there are different families of filtering algorithms. When F is linear and p_u is Gaussian, the analytical formula for $p(u_t|y_{1:t})$ can be obtained through the *Kalman filter* algorithm. When F is nonlinear or p_u is non-Gaussian, both predictive distribution and filtering distribution can be approximated by a number of particles. Equation (1.2.6) can be approached by importance sampling, which leads to the *particle filter* algorithm, or by approximating the prior $p(u_t|y_{1:t-1})$ with a Gaussian distribution, which leads to the *ensemble Kalman filter* algorithm.

Kalman Filter Algorithm When F is linear (i.e., $F \in \mathbb{R}^{d_u \times d_u}$) and p_u is Gaussian in Eq. (1.2.1), it is easy to derive that all predictive and filtering distributions are Gaussian. It remains to figure out their means and covariance matrices. Assume $p(u_{t-1}|y_{1:t-1}) = \mathcal{N}\big(u_{t-1}; m_{t-1}, C_{t-1}\big)$, with the convention $p(\cdot|y_{1:0}) := p(\cdot)$. It follows from Eq. (1.2.4) that the predictive distribution is

$$p(u_t|y_{1:t-1}) = \mathcal{N}(u_t; \widehat{m}_t, \widehat{C}_t), \tag{1.2.7}$$

$$\text{where} \quad \widehat{m}_t := F m_{t-1}, \tag{1.2.8}$$

$$\widehat{C}_t := F C_{t-1} F^\top + Q. \tag{1.2.9}$$

To compute the filtering distribution $p(u_t|y_{1:t})$, we can first derive from Eq. (1.2.6) that (u_t, y_t) is jointly Gaussian conditioning on $y_{1:t-1}$:

$$\begin{bmatrix} u_t|y_{1:t-1} \\ y_t|y_{1:t-1} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \widehat{m}_t \\ H_t\widehat{m}_t \end{bmatrix}, \begin{bmatrix} \widehat{C}_t & \widehat{C}_t H_t^\top \\ H_t\widehat{C}_t & H_t\widehat{C}_t H_t^\top + R_t \end{bmatrix} \right) \tag{1.2.10}$$

using the relationship Eq. (1.2.6). We can then apply the conditional Gaussian formula to obtain the filtering distribution $p(u_t|y_t, y_{1:t-1}) = p(u_t|y_{1:t})$:

$$p(u_t|y_{1:t}) = \mathcal{N}(u_t; m_t, C_t), \tag{1.2.11}$$

$$\text{where} \quad m_t := \widehat{m}_t + \widehat{K}_t(y_t - H_t\widehat{m}_t), \tag{1.2.12}$$

$$C_t := (I - \widehat{K}_t H_t)\widehat{C}_t, \tag{1.2.13}$$

$$\widehat{K}_t := \widehat{C}_t H_t^\top (H_t \widehat{C}_t H_t^\top + R_t)^{-1}. \tag{1.2.14}$$

Although the Kalman filter algorithm only applies to certain classes of state space models, it provides guidance to many other filtering algorithms, including the ensemble Kalman filter algorithm, which will be one of the main focuses of this thesis. Moreover, a side product of the computation Eq. (1.2.10) gives:

$$p(y_t|y_{1:t-1}) = \mathcal{N}(y_t; H_t\widehat{m}_t, H_t\widehat{C}_tH_t^\top + R_t), \tag{1.2.15}$$

which makes it possible to compute the *model evidence* $p(y_{1:T}) = \prod_{t=1}^T p(y_t|y_{1:t-1})$. This will be an important concept in Chapter 3 and Chapter 4.

Particle Filter Algorithm When F is nonlinear or p_u is non-Gaussian, we may not have explicit formulas for the predictive and filtering distributions. One way to approximate these distributions is to use weighted particles. Assume that $(w_{t-1}^{1:N}, u_{t-1}^{1:N})$ forms an approximation to the filtering distribution $p(u_{t-1}|y_{1:t-1})$, where $\sum_{i=1}^N w_{t-1}^i = 1$. Then the predictive distribution $p(u_t|y_{1:t-1})$ can be approximated with $(w_{t-1}^{1:N}, \widehat{u}_t^{1:N})$ where

$$\widehat{u}_t^n = F(u_{t-1}^n) + \xi_t^n, \qquad \xi_t^n \sim \mathcal{N}(0, Q), \tag{1.2.16}$$

due to the relationship Eq. (1.2.1). The inverse problem Eq. (1.2.6) for the filtering distribution $p(u_t|y_{1:t})$ can be solved via importance sampling, where each particle is reweighted with their corresponding data likelihood after self-normalization, i.e.:

$$w_t^n = \frac{\widehat{w}_t^n}{\sum_{i=1}^N \widehat{w}_t^i}, \qquad \text{where } \widehat{w}_t^n = w_{t-1}^n \mathcal{N}(y_t; H_t\widehat{u}_t^n, R_t), \tag{1.2.17}$$

and $u_t^n = \widehat{u}_t^n$. More advanced techniques like the resampling procedure and different choices of the proposal kernel can be found in §3.7.4, and we refer the reader to [Doucet and Johansen, 2009] for more details. Similar to importance sampling, $\sum_{i=1}^N \widehat{w}_t^i$ can be used to approximate the normalizing constant of the inverse problem Eq. (1.2.6), which is $p(y_t|y_{1:t-1})$. We will explore the application of particle filter algorithms to estimation of model evidence $p(y_{1:T})$ in Chapter 3.

Ensemble Kalman Filter Algorithm Another way to approximate the predictive and filtering distributions when F is nonlinear or p_u is non-Gaussian is to use Gaussian approximations. Assume that the equally weighted particles $u_{t-1}^{1:N}$ forms an approximation to the filtering distribution $p(u_{t-1}|y_{1:t-1})$. Then the predictive distribution $p(u_t|y_{1:t-1})$ can be approximated with $\widehat{u}_t^{1:N}$, defined in the same way as Eq. (1.2.16). Different from particle filter algorithms, the inverse problem Eq. (1.2.6) for the filtering distribution is solved via finding a Gaussian approximation of the prior/predictive distribution $p(u_t|y_{1:t-1})$ using the empirical moments of $\widehat{u}_t^{1:N}$:

$$\widehat{m}_t := \frac{1}{N}\sum_{n=1}^{N}\widehat{u}_t^n, \qquad \widehat{C}_t := \frac{1}{N-1}\sum_{n=1}^{N}(\widehat{u}_t^n - \widehat{m}_t)(\widehat{u}_t^n - \widehat{m}_t)^\top. \tag{1.2.18}$$

Then, the analysis step is defined similarly to the Kalman filter updates Eqs. (1.2.12) to (1.2.14), except that each individual particle is updated instead of the mean, and $\widehat{K}_t$ is computed using empirical moments in Eq. (1.2.18):

$$u_t^n := \widehat{u}_t^n + \widehat{K}_t(y_t + \eta_t^n - H_t u_t^n), \qquad \eta_t^n \sim \mathcal{N}(0, R_t), \tag{1.2.19}$$

$$\widehat{K}_t := \widehat{C}_t H_t^\top (H_t \widehat{C}_t H_t^\top + R_t)^{-1}. \tag{1.2.20}$$

Here η_t^n is introduced as a perturbation term to ensure that in linear Gaussian problems, the empirical moments of $u_t^{1:N}$ converge as $N \to \infty$ to the moments of the true filtering distribution $p(u_t|y_{1:t})$. In nonlinear settings, empirical results have shown that the ensemble Kalman filter algorithms are still able to achieve good reconstruction performance, even when u_t's are high-dimensional and the number of particles N is smaller than d_u.

Following a similar argument as in the Kalman filter section, we have the following approximation of $p(y_t|y_{1:t-1})$:

$$p(y_t|y_{1:t-1}) \approx \mathcal{N}(y_t; H_t\widehat{m}_t, H_t\widehat{C}_t H_t^\top + R_t), \tag{1.2.21}$$

where $\widehat{m}_t$ and $\widehat{C}_t$ are the empirical moments computed in Eq. (1.2.18). This makes it possible to approximate the model evidence $p(y_{1:T})$, even in high-dimensional and nonlinear problems. This concept will be explored further in Chapter 3 and Chapter 4.

1.3 Data-driven Inverse Problems: Motivations

In Section 1.1, we discussed algorithms for the inverse problem (1.1.1) where the forward map $\mathcal{A}$ is known explicitly. However, there are scenarios where: (1) we do not have perfect knowledge about $\mathcal{A}$; or (2) $\mathcal{A}$ is expensive to simulate, which limits the number of forward map evaluations that can be performed in the reconstruction algorithm. We provide an example for motivation:

Consider the Darcy flow inverse problem of reconstructing the PDE diffusion coefficient $u(x)$ from sparse observations of the field $p(x)$:

$$-\nabla \cdot (u(x)\nabla p(x)) = f(x), \qquad x \in D, \tag{1.3.1}$$

$$p(x) = 0, \qquad x \in \partial D. \tag{1.3.2}$$

Here $D = [0,1]^d$ and f is known. More specifically, the forward map $\mathcal{A}$ is defined by the composition of the PDE solver $u \mapsto p$ (which depends on the choice of f) and the pointwise evaluation map $p \mapsto \big(p(x_1), \dots, p(x_n)\big)^\top$ at some observation locations $x_1, \dots, x_n \in D$. Observation $y \in \mathbb{R}^n$ is obtained through a perturbation of $\big(p(x_1), \dots, p(x_n)\big)^\top$. Reconstruction of u from y can be achieved using optimization or MCMC techniques (see, e.g., [Huang et al., 2022; Garbuno-Inigo et al., 2020; Chada et al., 2020; Bigoni et al., 2019]).

Now consider the following scenarios:

1. f is not known perfectly. For instance, f is a piecewise constant function with unknown values at certain regions. Therefore, the PDE solver within $\mathcal{A}$ contains unknown parameters (as it depends on f). To solve the inverse problem for u from y, unknown

parameters of $\mathcal{A}$ also need to be inferred.

2. The PDE solver of $\mathcal{A}$ is computationally expensive, which is true for high dimensional cases or when a fine grid is needed to resolve small scales. We aim to find an efficient surrogate PDE solver that can approximate the original PDE solver within $\mathcal{A}$, for example, with an adaptive grid. The locations of grid points are treated as parameters of the surrogate model used to approximate $\mathcal{A}$. To solve the inverse problem for u from y, parameters of the surrogate model also need to be inferred.

Therefore, we aim to approximate $\mathcal{A}$ by $\mathcal{A}_\theta$, where θ denotes (1) unknown parameters of $\mathcal{A}$ that need to be identified; or (2) parameters of a surrogate model in replacement of $\mathcal{A}$ for a lower simulation cost. We propose data-driven approaches that jointly learn θ and reconstruct u, from observation data y alone.

In Section 1.2, we discussed algorithms for data assimilation problem (1.2.1)-(1.2.3) where the model is known perfectly, i.e., F, Q, H_t, R_t, p_u are known. However, there are scenarios where: (1) we do not have perfect knowledge about the model dynamics F, Q; or (2) F is expensive to simulate, which increases computational time of the reconstruction algorithm. We provide two examples for motivation:

1. Consider the Lorenz-63 model of simulating a 3-D chaotic system:

$$\frac{\mathrm{d}u}{\mathrm{d}s} = f(u), \qquad \begin{cases} f^{(1)}(u) = \sigma(u^{(2)} - u^{(1)}), \\ f^{(2)}(u) = u^{(1)}(\rho - u^{(3)}) - u^{(2)}, \\ f^{(3)}(u) = u^{(1)}u^{(2)} - \beta u^{(3)}. \end{cases} \tag{1.3.3}$$

Here σ, ρ, β are known (typical choices of values are $10, 28, 8/3$). $u^{(i)}$ and $f^{(i)}$ denote the i-th coordinate of u and component of f. Assume observations y_t are made directly from u every Δ_s second, with observation matrix H_t and Gaussian noise perturbation η_t. Therefore, F corresponds to the flow map of the ODE $u(s) \mapsto u(s+\Delta_s)$ and $Q = 0$.

Reconstruction of u_t's from y_t's can be achieved using filtering algorithms (see, e.g., [Evensen et al., 2022]).

It is often the case that our knowledge of the dynamical system is imperfect, for instance, when the exact values of σ, ρ, β are unknown. Then we do not have perfect knowledge about F. To solve the data assimilation problem for u_t's from y_t's, unknown parameters of F (which in this case are σ, ρ, β) also need to be inferred.

2. Consider the Kuramoto-Sivashinsky (KS) model of simulating a high-dimensional chaotic system:
$$\frac{\partial u}{\partial s} = -\nu \frac{\partial^4 u}{\partial x^4} - \frac{\partial^2 u}{\partial x^2} - u \frac{\partial u}{\partial x}. \tag{1.3.4}$$
Here $u(x, s)$ is defined on $[0, L] \times [0, \infty)$ with suitable initial and boundary conditions, and the viscosity parameter ν is known. Assume observations y_t's are made directly from the function $u(\cdot, s)$ every Δ_s second, at certain observation locations $x_1, \ldots, x_n \in [0, L]$, with observation matrix H_t and Gaussian noise perturbation η_t. Then F corresponds to the map $u(\cdot, s) \mapsto u(\cdot, s + \Delta_s)$. More discussion on this setup and the discretization of F can be found in Chapter 4. Reconstruction of u_t's from y_t's can be achieved using filtering algorithms (see, e.g., [Jardak et al., 2010]).

   An accurate simulation of F is computationally expensive, as it requires discretization over a very fine grid. This increases the cost of the data assimilation algorithm, as it requires simulating F at each assimilation step. Therefore, we would like to find an efficient surrogate model that can approximate F with good accuracy, for example, with a neural network. To solve the data assimilation problem for u_t's from y_t's, parameters of the surrogate model also need to be inferred.

Therefore, we aim to approximate F, Q by F_α, Q_β, where $\theta := \{\alpha, \beta\}$ denotes (1) unknown parameters of the model dynamics that need to be identified; or (2) parameters of a surrogate model in replacement of F, Q for a lower computational cost. If u_t is high-dimensional, we can also approximate (1.2.1) using a reduced-order model with low-dimensional surrogate dynamics, parameterized by θ. We propose data-driven approaches that jointly learn θ and reconstruct u_t's, from observations y_t's alone.

1.4 Outline and Main Contributions

We give an outline of the following chapters and summarize their main contributions.

- Chapter 3 is developed based on [Bigoni et al., 2020]. We focus on the IP setting (1.1.1), where the forward model $\mathcal{A}$ is defined as a differential equation solver, which is often expensive or hard to simulate. We approximate $\mathcal{A}$ by a numerical solver $\mathcal{A}_\theta$, where θ denotes the parameter governing the discretization, e.g., location of the grid. The main idea is to treat θ as part of the unknown, and jointly estimate θ and u with MCMC algorithms. We numerically show that in a variety of inverse problems arising in mechanical engineering, signal processing and the geosciences, the observations y contain useful information to guide the choice of discretization.

- Chapter 4 is developed based on [Chen et al., 2022]. We focus on the DA setting (1.2.1)-(1.2.3) when the state u_t is high-dimensional and both F, Q in the model dynamics (1.2.1) are unknown, approximated by F_α, Q_β. We introduce a machine learning framework, named autodifferentiable ensemble Kalman filter (AD-EnKF), that jointly learns the parameter $\theta := (\alpha, \beta)$ in the model dynamics and reconstructs the states u_t's. We leverage the ability of EnKF to scale to high-dimensional states, and the power of automatic differentiation to train high-dimensional surrogate models of the dynamics. We numerically show that, in the Lorenz-96 model, AD-EnKF outperforms existing methods that use expectation-maximization or particle filters to merge data assimilation and machine learning.

- Chapter 5 is developed based on [Chen et al., 2023]. We focus on the DA setting

(1.2.1)-(1.2.3) when the state u_t is high-dimensional and the model dynamics (1.2.1) are unknown. We build a surrogate model for the dynamics in a low-dimensional latent space, along with a decoder from latent space to state space. We propose a machine learning framework, named reduced-order autodifferentiable ensemble Kalman filter (ROAD-EnKF), that jointly learns the parameter θ in the surrogate model as well as the decoder, and reconstructs the states u_t's. In so doing, we blend the ideas of data assimilation, reduced-order modeling and automatic differentiation. We numerically show that, compared to existing methods, ROAD-EnKF achieves higher or comparable accuracy at a lower computational cost, making them a promising approach for surrogate state reconstruction and forecasting.

- Chapter 6 is developed based on [Chada et al., 2021]. We focus on the IP setting (1.1.1), where we assume the model is known perfectly. We review a family of derivate-free methods for reconstruction of u, namely iterative ensemble Kalman methods, that blends the ideas of ensemble data assimilation and least-squares optimization. We identify, compare and develop three subfamilies of ensemble methods that differ in the objective they seek to minimize and the derivative-based optimization scheme they approximate through the ensemble. We emphasize two principles for the derivation and analysis of iterative ensemble Kalman methods: statistical linearization and continuum limits. Following these guiding principles, we introduce new iterative ensemble Kalman methods that show promising numerical performance in Bayesian inverse problems, data assimilation and machine learning tasks.

CHAPTER 2
DATA-DRIVEN FORWARD DISCRETIZATIONS FOR BAYESIAN INVERSION

2.1 Introduction

Models used in science and engineering are often described by problem-specific input parameters that are estimated from indirect and noisy observations. The inverse problem of input reconstruction is defined in terms of a *forward model* from inputs to observable quantities, which in many applications needs to be approximated by discretization. A broad class of examples motivating this paper is the reconstruction of input parameters of differential equations. The choice of forward model discretization is particularly important in Bayesian formulations of inverse problems: discretizations need to be cheap since statistical recovery may involve millions of evaluations of the discretized forward model; they also need to be accurate enough to enable input reconstruction. The goal of this paper is to suggest a simple data-driven framework to build forward model discretizations to be used in Bayesian inverse problems. The resulting discretizations are data-driven in that they finely resolve regions of the input space where the data are most informative, while keeping the cost moderate by coarsely resolving regions that are not informed by the data. To be concrete and explain the idea, let us consider the inverse problem of recovering an unknown u from data y related by

$$y = \mathcal{G}(u) + \eta, \tag{2.1.1}$$

where $\mathcal{G}$ denotes the forward model from inputs to observables, $\eta \sim N(0, \Gamma)$ represents model error and observation noise, and Γ denotes a positive definite noise covariance matrix. We will follow a Bayesian approach, viewing u as a random variable [Kaipio and Somersalo, 2006; Stuart, 2010; Sanz-Alonso et al., 2018] with *prior* distribution $p_u(u)$. The Bayesian solution to the inverse problem is the *posterior* distribution $p_{u|y}(u)$ of u given the data y, which by an informal application of Bayes theorem is characterized by

$$p_{u|y}(u) \propto \exp\bigl(-\Phi(u;y)\bigr)p_u(u), \qquad \Phi(u;y) := \frac{1}{2}\left\|y - \mathcal{G}(u)\right\|_\Gamma^2 \tag{2.1.2}$$

with $\|\cdot\|_\Gamma := \|\Gamma^{-1/2}\cdot\|$. A common computational bottleneck arises when the forward model $\mathcal{G}$ and hence the likelihood are intractable, meaning that it is impossible or too costly to evaluate. This paper introduces a framework to tackle this computational challenge by employing data-driven discretizations of the forward model. The main idea is to include the parameters that govern the discretization as part of the unknown to be estimated within the Bayesian machinery. More precisely, we consider a family $\{\mathcal{G}^a\}_{a\in\mathcal{A}}$ of approximate forward models and put a prior $q_{u,a}(u,a)$ over both unknown inputs u and forward discretization parameters $a \in \mathcal{A}$ to define a joint posterior

$$q_{u,a|y}(u,a) \propto \exp\bigl(-\Psi(u,a)\bigr)q_{u,a}(u,a), \qquad \Psi(u,a;y) := \frac{1}{2}\left\|y - \mathcal{G}^a(u)\right\|_\Gamma^2. \tag{2.1.3}$$

While this structure underlies many hierarchical formulations of Bayesian inverse problems [Kaipio and Somersalo, 2006], in this paper the hyper-parameter a determines the choice of discretization of the forward model $\mathcal{G}$.

Including the learning of the numerical discretizations of the forward map as part of the inference agrees with the Bayesian philosophy of treating unknown quantities as random variables, and is also in the spirit of recent probabilistic numerical methods [Cockayne et al., 2019]; rather than implicitly assuming that a true hidden numerical discretization of the forward model generates the data, a Bayesian would acknowledge the uncertainty in the choice of a suitable discretization and let the observed data inform such a choice. Moreover, the Bayesian viewpoint has two main practical advantages. First, data-informed grids will typically be coarse in regions of the input space that are not informed by the data, allowing successful input reconstruction at a reduced computational cost. Second, the posterior $q_{u,a|y}(u, a)$ contains useful uncertainty quantification on the discretizations. This additional uncertainty information may be exploited to build a high-fidelity forward model to be employed within existing inverse problem solvers, either in Bayesian or classical settings.

2.1.1 Related Work

The Bayesian formulation of inverse problems provides a flexible and principled way to combine data with prior knowledge. However, in practice it is rarely possible to perform posterior inference with the model of interest (2.1.2) due to various computational challenges. In this paper we investigate the construction of computable data-driven forward discretizations of intractable likelihoods arising in the inversion of differential equations. Other intertwined obstacles for posterior inference are:

- **Sampling cost.** While exact posterior inference is often intractable, approximate posterior inference can be performed by employing sampling algorithms. Markov chain Monte Carlo and particle-based methods are popular, but implementations of these algorithms require repeated evaluation of the forward model $\mathcal{G}$, which may be costly.

- **Large input dimension.** The unknown parameter u may be high, or even infinite dimensional. While the convergence rate of certain sampling schemes may be independent of the input dimension [Cotter et al., 2013; Agapiou et al., 2017; Garcia Trillos et al., 2017], the computational and memory cost per sample may increase prohibitively with dimension.

- **Model error.** The forward model is only an approximation of the real relationship between input and observable output variables. Model discrepancy can damage input recovery.

- **Complex geometry.** The unknown may be a function defined on a complex, perhaps unknown domain that needs to be approximated.

All these challenges have long been identified [Sacks et al., 1989; Kennedy and O'Hagan, 2001; Kaipio and Somersalo, 2006, 2007], giving rise to a host of methods for sampling, parameter reduction, model reduction, enhanced model error techniques and geometric methods for inverse problems. We focus on the model-reduction problem of building forward discretizations, but the methodology proposed in this paper can be naturally combined with existing techniques that address complementary challenges. For instance, our forward model discretizations may be used within multilevel MCMC methods [Giles, 2008b] or within two-stage sampling methods [Green and Mira, 2001; Tierney and Mira, 1999; Christen and Fox, 2005; Cui et al., 2015; Efendiev et al., 2006], and thus help to reduce the sampling cost. Also, forward model discretizations may be combined with parameter reduction and model adaptation techniques, as in [Lieberman et al., 2010; Li et al., 2018]. It is important, however, to distinguish between the parameter and model reduction problems. While the former aims to find suitable small-dimensional representations of the input u, the latter is concerned with effectively reducing the number of degrees of freedom used to compute the forward model $\mathcal{G}$. In regards to model error, our framework may be thought of as incorporating Bayesian model choice to the Bayesian solution of inverse problems by viewing each forward model discretization as a potential model. Following this interpretation, the *a posteriori* choice of forward discretization may in principle be determined using Bayes factors. Lastly, learning appropriate discretizations of forward models is particularly important for inverse problems set in complex, possibly uncertain geometries [Garcia Trillos et al., 2017; Garcia Trillos and Sanz-Alonso, 2018; Harlim et al., 2020].

Many approaches to computing forward map surrogates and reduced-order models have been proposed; we refer to [Frangos et al., 2010] for an extended survey, and to [Peherstorfer et al., 2018] for a broader discussion of multi-fidelity models in other outer-loop applications.

Most methods fall naturally into one of three categories:

1. Projection-based methods: the forward model equations are described in a reduced basis that is constructed using few high-fidelity forward solves (called snapshots). Two popular ways to construct the reduced basis are proper orthogonal decomposition (POD) and reduced order basis. In the inverse problem context, data-informed construction of snapshots [Cui et al., 2015] allows to approximate the posterior support with fewer high-fidelity forward runs. To our knowledge, there is little theory to guide the required number or location of snapshots to meet a given error tolerance.

2. Spectral methods: polynomial chaos [Xiu and Karniadakis, 2002] is a popular method for forward propagation of uncertainty, that has more recently been used to produce surrogates for intractable likelihoods [Marzouk et al., 2007]. The paper [Marzouk and Xiu, 2009] translates error in the likelihood approximation to Kullback-Leibler posterior error. A drawback of these methods is that they are only practical when the random inputs can be represented by a small number of random variables. Recent interest lies in adapting the spectral approximations to observed data [Li and Marzouk, 2014].

3. Gaussian processes and neural networks: some of the earliest efforts to allow for Bayesian inference with complex models suggested to use Gaussian processes [Rasmussen and Williams, 2006] to construct surrogate likelihood models [Sacks et al., 1989; Kennedy and O'Hagan, 2001]. The accuracy of the resulting approximations has been studied in [Stuart and Teckentrup, 2017], which again requires a suitable representation of the input space. Finally, representation of the likelihood using neural networks in combination with generalized polynomial chaos expansions has been investigated in [Schwab and Zech, 2019].

This paper focuses on grid-based discretizations and density-based discretizations of static inverse problems arising in mechanical engineering, signal processing and the geophysical sciences. However, the proposed framework may be used in conjunction with other reduced-order models, in dynamic data assimilation problems, and in other applications. Finally, we mention that for classical formulations of certain specific inverse problems, optimal forward discretization choices have been proposed [Borcea et al., 2005; Becker and Vexler, 2005].

2.1.2 Outline and Contributions

Section 2.2 reviews the Bayesian formulation of inverse problems. Section 2.3 describes the main framework for the Bayesian learning of forward map discretizations. We will consider two ways to parametrize discretizations: in the first, the grid points locations are learned directly, and in the second we learn a probability density from which to obtain the grid. In Section 2.4 we discuss a general approach to sampling the joint posterior over unknown input and discretization parameters, which consists of a Metropolis-within-Gibbs that alternates between a reversible jump Markov chain Monte Carlo (MCMC) algorithm to update the discretization parameters and a standard MCMC to update the unknown input. Section 2.5 demonstrates the applicability, benefits, and limitations of our approach in a variety of inverse problems arising in mechanical engineering, signal processing and source detection, considering Euler discretization of ODEs, Euler-Maruyama discretization of SDEs, and finite element methods for PDEs. We conclude in Section 2.6 with some open questions for further research.

2.2 Background: Bayesian Formulation of Inverse Problems

Consider the inverse problem of recovering an unknown $u \in \mathsf{U}$ from data $y \in \mathbb{R}^m$ related by

$$y = \mathcal{G}(u) + \eta, \tag{2.2.1}$$

where U is a space of admissible unknowns and η is a random variable whose distribution is known to us, but not its realization. In many applications, the *forward model* $\mathcal{G} : \mathsf{U} \to \mathbb{R}^m$ can be written as the composition of forward and observation maps, $\mathcal{G} = \mathcal{O} \circ \mathcal{F}$. The forward map $\mathcal{F} : \mathsf{U} \to \mathbb{Z}$ represents a *complex* mathematical model that assigns outputs $z \in \mathbb{Z}$ to inputs $u \in \mathsf{U}$. For instance, u may be the parameters of a differential equation, and z may be its analytical solution. The observation map $\mathcal{O} : \mathbb{Z} \to \mathcal{Y}$ establishes a link between outputs and observable quantities, e.g. by point-wise evaluation of the solution.

In the Bayesian formulation of the inverse problem (2.2.1), one specifies a *prior* distribution on u and seeks to characterize the *posterior* distribution of u given y. If the input space U is finite dimensional, $\mathsf{U} \subset \mathbb{R}^d$, then the prior distribution, denoted as $p_u(u)$, can be defined through its Lebesgue density. The noise distribution of η in $\mathbb{R}^m$ gives the *likelihood* $p_{y|u}(y|u)$. In this work we assume, for concreteness, that η is a zero-mean Gaussian with covariance $\Gamma \in \mathbb{R}^{m\times m}$, so that

$$p_{y|u}(y|u) \propto \exp\bigl(-\Phi(u;y)\bigr), \qquad \Phi(u;y) := \frac{1}{2}\left\|y - \mathcal{G}(u)\right\|_\Gamma^2, \tag{2.2.2}$$

where $\|\cdot\|_\Gamma := \|\Gamma^{-1/2}\cdot\|$. Using Bayes' formula, the posterior density is given by

$$p_{u|y}(u) = \frac{1}{Z}\, p_{y|u}(y|u)p_u(u), \qquad Z = \int_{\mathsf{U}} p_{y|u}(y|u)p_u(u)\mathrm{d}u \tag{2.2.3}$$

with multiplicative constant Z depending on y.

For many inverse problems of interest, the unknown u is a function and the input space U is an infinite-dimensional Banach space. In such a case, the prior cannot be specified in terms of its Lebesgue density, but rather as a measure μ_u supported on U. Provided that $\mathcal{G} : \mathsf{U} \to \mathbb{R}^m$ is measurable and that $\mu_u(\mathsf{U}) = 1$, the posterior measure $\mu_{u|y}$ is still defined, in analogy to (2.2.3), as a change of measure with respect to the prior

$$\frac{\mathrm{d}\mu_{u|y}}{\mathrm{d}\mu_u}(u) \propto \exp\bigl(-\Phi(u;y)\bigr). \tag{2.2.4}$$

We refer to [Stuart, 2010] and [García Trillos and Sanz-Alonso, 2017] for further details. The posterior $\mu_{u|y}$ contains, in a precise sense [Zellner, 1988], all the information on u available in the data y and the prior μ_u. This paper is concerned with inverse problems where $\mathcal{G} = \mathcal{O} \circ \mathcal{F}$ arises from a complex model $\mathcal{F}$ that cannot be evaluated pointwise; we then seek to approximate the *idealized* posterior $\mu_{u|y}$ finding a compromise between accuracy and computational cost.

A simple but important observation is that approximating $\mathcal{F}$ accurately is *not* necessary in order to approximate $\mu_{u|y}$ accurately. It is *only* necessary to approximate $\mathcal{G} = \mathcal{O} \circ \mathcal{F}$, since $\mathcal{F}$ appears in the posterior only through $\mathcal{G}$. While producing discretizations to complex models $\mathcal{F}$ has been widely studied in numerical analysis, here we investigate how to approximate $\mathcal{F}$ with the specific goal of approximating the posterior $\mu_{u|y}$, incorporating prior and data knowledge into the discretizations. For some inverse problems the observation operator $\mathcal{O}$ also needs to be discretized, leading to similar considerations.

2.3 Bayesian Discretization of the Forward Model

Suppose that $\mathcal{F}$ is the solution map to a differential equation that cannot be solved in closed form, and $\mathcal{O}$ is point-wise evaluation of the solution. Standard practice in computing the Bayesian solution to the inverse problem involves using an *a priori* fixed discretization, e.g., by discretizing the domain of the differential equation into a fine grid. Provided that the grid is fine enough, the posterior defined with the discretized forward map can approximate well the one in (2.2.4). However, the discretizations are usually performed on a *fine* uniform grid which may lead to unnecessary waste of computational resources. Indeed, it is expected that the choice of discretization should be problem dependent, and should be informed both by the observation locations (which are often not uniform in space) and by the value of the unknown input parameter that we seek to reconstruct. Thus we seek to learn *jointly* the unknown input u and the discretization of the forward map.

We will consider a parametric family of discretizations. Precisely, we let

$$\mathcal{A} := \Big\{ a = (k, \theta) : \, k \in \mathsf{K} \subset \{1, 2, \ldots\}, \, \theta \in D(k) \subset \mathbb{R}^{d(k)} \Big\}, \tag{2.3.1}$$

and each pair $a = (k, \theta) \in \mathcal{A}$ will parameterize a discretized forward model $\mathcal{G}^a$. For given $k \in \mathsf{K}$, $d(k)$ represents the degrees of freedom in the discretization, and $\theta \in D(k)$ is the $d(k)$-dimensional model parameter of the discretization, where $D(k)$ is the region containing all parameters of interest. In analogy with Bayesian model selection frameworks [Robert, 2007; Green, 1995], $k \in \mathsf{K}$ may be interpreted as indexing the discretization model. We focus on the model reduction rather than the parameter reduction problem, and assume that all approximation maps share the same input and output spaces U and $\mathbb{Z}$. We will illustrate the flexibility of this framework using grid-based approximations and density-based discretizations.

Example 2.3.1 (Grid-based discretizations)**.** *Here the first component of each element $a = (k, \theta) \in \mathcal{A}$ represents the number of points in a grid. The set K contains all allowed grid sizes. If we denote $D \subset \mathbb{R}^d$ as the temporal or spatial domain of the equation being discretized, we define $D(k) = D^k$ and $d(k) = d \times k$, where $D^k := D \times \cdots \times D$ denotes the k-fold Cartesian product of D. Then the second component $\theta = [x_1, \ldots x_k]$ encodes the locations of k grid points.*

Example 2.3.2 (Density-based discretizations)**.** *Here the first component of each element $a = (k, \theta) \in \mathcal{A}$ represents again the number of points in a grid, and the second component parametrizes a probability density $\rho = \rho(x; \theta)$ on the temporal or spatial domain of interest,* *by a parameter θ of fixed dimension, independent of k. Given $a \in \mathcal{A}$ we may for instance employ MacQueen's method [Du and Gunzburger, 2002] to formulate a centroidal Voronoi tessellation, which outputs k generators $\{x_1, \ldots, x_k\}$, and then use them as grid points to generate a finite element grid by Delaunay triangulation. Intuitively θ controls the spatial density of the non-uniform grid points $\{x_1, \ldots, x_k\}$. The space K represents, as before, all the allowed number of grid points.*

Example 2.3.3 (Other discretizations)**.** *As mentioned in the introduction, other discretizations and model reduction techniques could be considered within the above framework, including projection-based approximations, Gaussian processes, and graph-based methods. However, in our numerical experiments we will focus on grid-based and density-based discretizations.*

We consider a product prior on $(u, a) \in \mathsf{U} \times \mathcal{A}$, given by

$$q_{u,a}(u, a) = q_u(u) q_a(a), \tag{2.3.2}$$

where $q_u(u) = p_u(u)$ is as in the original, idealized inverse problem (2.2.1). In general, conditioning on u may or may not provide useful information about how to approximate $\mathcal{G}(u)$. When it does, this can be infused into the prior by letting the conditional distribution of a given u depend on u. For simplicity we restrict ourselves to the product structure (2.3.2).

The examples above and the structure of the space $\mathcal{A}$ defined in equation (2.3.1) suggest to define hierarchically a prior over $a \in \mathcal{A}$

$$q_a(a) = q_{k,\theta}(k, \theta) = q_k(k) q_{\theta|k}(\theta|k), \tag{2.3.3}$$

where $q_k(k)$ is a probability mass function that penalizes expensive discretizations that employ large number $d(k)$ of degrees of freedom, and $q_{\theta|k}(\theta|k)$ denotes the conditional distribution of θ given k in $D(k)$.

We define the likelihood of observing data y given (u, a) by

$$q_{y|u,a}(y|u,a) \propto \exp\bigl(-\Psi(u,a;y)\bigr), \qquad \Psi(u,a;y) := \frac{1}{2}\left\|y - \mathcal{G}^a(u)\right\|_\Gamma^2, \tag{2.3.4}$$

where $\mathcal{G}^a = \mathcal{O} \circ \mathcal{F}^a$. The discretized forward maps $\mathcal{F}^a$ will be chosen so that evaluating Ψ is possible.

We first consider the case where $\mathsf{K} = \{k\}$ is a singleton, and $\mathcal{A} := \Big\{a = (k, \theta) : \theta \in D(k) \subset \mathbb{R}^{d(k)}\Big\}$ has a Euclidean space structure. Then, by Bayes' formula,

$$q_{u,a|y}(u,a) = \frac{1}{\tilde{Z}} q_{y|u,a}(y|u,a) q_u(u) q_a(a), \qquad \tilde{Z} = \int_{\mathsf{U}\times\mathcal{A}} q_{y|u,a}(y|u,a) q_u(u) q_a(a) \mathrm{d}u\mathrm{d}a, \tag{2.3.5}$$

where $\tilde{Z} = \tilde{Z}(y)$ is a normalizing constant. The first marginal of $q_{u,a|y}(u,a)$, which we denote as $q_{u|y}(u)$, constitutes a data-informed approximation of the posterior $p_{u|y}(u)$ from the full, idealized inverse problem (2.2.3). We have the following result:

Proposition 2.3.4. *Let $p_{u|y}(u)$ be defined as in (2.2.3) and $q_{u|y}(u)$ be defined as above. If $\mathcal{G}$ is bounded and, for $q_{u,a}$-almost any (u,a), $\|\mathcal{G}^a(u) - \mathcal{G}(u)\| < \epsilon$, then*

$$d_{TV}\bigl(q_{u|y}(u), p_{u|y}(u)\bigr) < C\epsilon \tag{2.3.6}$$

for some constant C independent of ϵ.

Proof. Integrating both sides of the first equation in (2.3.5) with respect to a:

$$q_{u|y}(u) = \frac{1}{\tilde{Z}}\Big(\int_{\mathcal{A}} q_{y|u,a}(y|u,a) q_a(a) \mathrm{d}a\Big) q_u(u) =: \frac{1}{\tilde{Z}} \tilde{g}_y(u) q_u(u).$$

Compare this to equation (2.2.3) and write $g_y(u) = p_{y|u}(y|u)$, we have

$$\|\tilde{g}_y(u) - g_y(u)\| \le \int_{\mathcal{A}} \exp\left(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2\right) - \exp\left(-\frac{1}{2}\|y - \mathcal{G}(u)\|_\Gamma^2\right) \mathrm{d}a \le C\|\mathcal{G}^a(u) - \mathcal{G}(u)\|,$$

where the last inequality follows from the Lipschitz continuity of e^{-w} for $w \geq 0$, boundedness of $\mathcal{G}$, and equivalence of norm in $\mathbb{R}^m$. This implies that $|\tilde{g}_y(u) - g_y(u)| \leq C\epsilon$ and hence $|\tilde{Z} - Z| \leq C\epsilon$. Then the statement follows from a slight modification of Theorem 1.14 in [Sanz-Alonso et al., 2018] and the definition of TV distance. $\square$

Now we are ready to extend the above results to infinite-dimensional input space U. We define a prior measure on $\mathsf{U} \times \mathcal{A}$ given by $\nu_{u,a}(du, da) = \nu_u(du) \times \nu_a(da)$, where $\nu_u(du) = \mu_u(du)$ is as in the idealized inverse problem. The posterior measure on $\mathsf{U} \times \mathcal{A}$ conditioning on y will still be denoted by $\nu_{u,a|y}$.

Proposition 2.3.5. *Suppose that* U *is a separable Banach space with* $\nu_u(\mathsf{U}) = 1$*,* $\Psi : \mathsf{U}\times\mathcal{A} \to \mathbb{R}$ *is continuous. Then the posterior measure* $\nu_{u,a|y}$ *of* (u, a) *given* y *is absolutely continuous with respect to the prior* $\nu_{u,a}$ *on* $\mathsf{U} \times \mathcal{A}$ *and has Radon-Nikodym derivative*

$$\frac{d\nu_{u,a|y}}{d\nu_{u,a}}(u, a) \propto \exp\big(-\Psi(u, a; y)\big). \tag{2.3.7}$$

Proof. By the disintegration theorem (which holds for arbitrary Radon measures on separable metric spaces –see [Dellacherie and Meyer, 2011] chapter 3, page 70) for all measurable subsets $U' \subseteq \mathsf{U}$, $A' \subseteq \mathcal{A}$ and $Y' \subseteq \mathcal{Y}$, we can write $\nu_{u,a,y}(U' \times A' \times Y')$ in two different ways:

$$\int_{Y'} \nu_{u,a|y}(U' \times A'|y)\mathrm{d}\nu_y(y) = \nu_{u,a,y}(U', A', Y') = \int_{U'\times A'} \nu_{y|u,a}(Y'|u, a)\mathrm{d}\nu_{u,a}(u, a).$$

In particular,

$$\nu_y(Y') = \int_{\mathsf{U}\times\mathcal{A}} \nu_{y|u,a}(Y'|u, a)\mathrm{d}\nu_{u,a}(u, a) = \int_{\mathsf{U}\times\mathcal{A}} \int_{Y'} Z_\Gamma^{-1} \exp\left(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2\right) \mathrm{d}y\mathrm{d}\nu_{u,a}(u, a),$$

given our assumptions on the noise model, where Z_Γ is a constant depending on the noise covariance Γ. We can then use Tonelli's theorem to swap the order of the integrals and obtain

$$\nu_y(Y') = \int_{Y'} \left(\int_{\mathsf{U}\times\mathcal{A}} Z_\Gamma^{-1} \exp\left(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2 \right) d\nu_{u,a}(u,a) \right) \mathrm{d}y.$$

Given that Y' is arbitrary, we conclude that ν_y is absolutely continuous with respect to the Lebesgue measure with density:

$$\frac{\mathrm{d}\nu_y(y)}{\mathrm{d}y} = \int_{\mathsf{U}\times\mathcal{A}} Z_\Gamma^{-1} \exp\left(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2 \right) \mathrm{d}\nu_{u,a}(u,a).$$

On the other hand,

$$\begin{aligned}
&\int_{U'\times A'} \nu_{y|u,a}(Y'|u,a)\mathrm{d}\nu_{u,a}(u,a) \\
&= \int_{U'\times A'} \int_{Y'} Z_\Gamma^{-1} \exp\left(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2 \right) \mathrm{d}y\mathrm{d}\nu_{u,a}(u,a) \\
&= \int_{U'\times A'} \int_{Y'} Z_\Gamma^{-1} \exp\left(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2 \right) \left(\frac{\mathrm{d}\nu_y(y)}{\mathrm{d}y} \right)^{-1} \mathrm{d}\nu_y(y)\mathrm{d}\nu_{u,a}(u,a) \\
&= \int_{Y'} \left(\int_{U'\times A'} Z_\Gamma^{-1} \exp\left(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2 \right) \left(\frac{\mathrm{d}\nu_y(y)}{\mathrm{d}y} \right)^{-1} \mathrm{d}\nu_{u,a}(u,a) \right) \mathrm{d}\nu_y(y),
\end{aligned}$$

applying Tonelli's theorem once again to obtain the last equality. Since Y' was arbitrary, it follows that for ν_y-a.e. y we have

$$\nu_{u,a|y}(U' \times A'|y) = \int_{U'\times A'} Z_\Gamma^{-1} \exp\left(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2 \right) \left(\frac{\mathrm{d}\nu_y(y)}{\mathrm{d}y} \right)^{-1} \mathrm{d}\nu_{u,a}(u,a).$$

In turn, from the arbitrariness of U', A' it follows that, for ν_y-a.e. y the measure $\nu_{u,a|y}(\cdot|y)$ is absolutely continuous with respect to $\nu_{u,a}$ and its Radon-Nykodym derivative satisfies

$$\frac{\mathrm{d}\nu_{u,a|y}}{\mathrm{d}\nu_{u,a}}(u,a) \propto \exp\bigl(-\frac{1}{2}\|y - \mathcal{G}^a(u)\|_\Gamma^2\bigr)$$

as claimed, where the constant of proportionality depends on y.

□

As in the finite dimensional case, we have the following result:

Proposition 2.3.6 (Well-posedness of Posterior)**.** *Under the same assumption as in Proposition 2.3.5, suppose further that for $q_{u,a}$-almost any (u, a), $\|\mathcal{G}^a(u) - \mathcal{G}(u)\| < \epsilon$, and $\mathcal{G}$ is bounded. Then we have*

$$d_{TV}(\nu_{u|y}, \mu_{u|y}) < C\epsilon \tag{2.3.8}$$

for some constant C independent of ϵ.

Remark 2.3.7. *In the context of grid-based forward approximations, the condition '$\|\mathcal{G}^a(u) - \mathcal{G}(u)\| < \epsilon$ $q_{u,a}$-almost surely' can be interpreted as 'almost any draw from the approximation parameter space $\mathcal{A}$ can produce an approximation of the forward model with error at most ϵ'. This is often the case, for example, when the grids are finer than some threshold under regularity conditions on the input space.*

2.4 Sampling the Posterior

The structure of the joint posterior $\nu_{u,a|y}$ over unknowns $u \in \mathsf{U}$ and approximations $a \in \mathcal{A}$ suggests using a Metropolis-within-Gibbs sampler, which constructs a Markov chain $(u^{(n)}, a^{(n)})$ by alternatingly sampling each coordinate:

Algorithm 2.4.1 Metropolis-within-Gibbs Core Structure

```
Choose $(u^{(1)}, a^{(1)}) \in \mathsf{U} \times \mathcal{A}$.
for $n = 1 : N$ do
    1. Sample $u^{(n+1)} \sim \mathbb{K}^{a^{(n)},y}(u^{(n)}|\cdot)$.
    2. Sample $a^{(n+1)} \sim \mathbb{L}^{u^{(n+1)},y}(a^{(n)}|\cdot)$.
end for
```

In the above, $\mathbb{K}^{a,y}$ and $\mathbb{L}^{u,y}$ are Metropolis-Hastings Markov kernels that are reversible with respect to $u|(a, y)$ and $a|(u, y)$. We remark that the kernel $\mathbb{K}^{a,y}$ involves evaluation of the forward model approximation $\mathcal{G}^a$ but not of the intractable full model $\mathcal{G}$. While the choice and design of the kernels $\mathbb{K}^{a,y}$ and $\mathbb{L}^{u,y}$ will clearly be problem-specific, and here we consider a standard method appropriate for the case where the input space U is a space of functions to define $\mathbb{K}^{a,y}$.

Before describing how to sample the full conditionals $\nu_{u|a,y}$ and $\nu_{a|u,y}$ of $u|(a,y)$ and $a|(u,y)$ it is useful to note that they satisfy the following expressions:

$$\frac{\mathrm{d}\nu_{u|a,y}}{\mathrm{d}\nu_u}(u) \propto \exp\Big(-\Psi(u,a;y)\Big), \quad \frac{\mathrm{d}\nu_{a|u,y}}{\mathrm{d}\nu_a}(a) \propto \exp\Big(-\Psi(u,a;y)\Big). \tag{2.4.1}$$

2.4.1 Sampling the Full Conditional $u|y,a$

For given a and y, we can sample from $\nu_{u|a,y}$ using pCN [Beskos et al., 2008], with proposal

$$\tilde{u} := \sqrt{1-\beta^2}u + \beta\xi, \qquad \xi \sim \mu_u,$$

and acceptance probability

$$\alpha(u,\tilde{u}) := \min\Big\{1, \exp\big(-\Psi(\tilde{u},a;y) + \Psi(u,a;y)\big)\Big\}.$$

Other discretization-invariant MCMC samplers [Cui et al., 2016; Rudolf and Sprungk, 2015] could also be used to update $u|y,a$, but pCN is a straightforward and effective choice in the examples considered here.

2.4.2 Sampling the Full Conditional $a|y,u$

Sampling Grid-based Discretizations

We will use a Markov kernel $\mathbb{L}^{u,y}(a|\cdot)$ written as a mixture of two kernels, i.e.

$$\mathbb{L}^{u,y}(a|\cdot) = \zeta\mathbb{L}_1^{u,y}(a|\cdot) + (1-\zeta)\mathbb{L}_2^{u,y}(a|\cdot),$$

each of which is induced by a different Metropolis-Hastings algorithm, and ζ determines the mixture weight. The proposal mechanism for each of the kernels corresponds to a different type of movement, described next:

1. For $\mathbb{L}_1^{u,y}(a|\cdot)$ we use Metropolis-Hastings to sample from the distribution $\nu_{a|u,y}$ using the following proposal: given $a = (k, \theta)$ with $\theta = [\theta_1, \dots, \theta_k]$ we set $\tilde{k} = k$ (i.e. the number of grid points stays the same) and let $\tilde{\theta}$ be defined by

$$\tilde{\theta}_i = \theta_i, \quad i = 1, \dots, k-1,$$

   and sample $\tilde{\theta}_k$ from a distribution on D with density (w.r.t. Lebesgue measure on D) τ_θ. In principle the density used to sample $\tilde{\theta}_k$ may depend on θ.

2. For $\mathbb{L}_2^{u,y}(a|\cdot)$ we use Metropolis-Hastings to sample from the distribution $\nu_{a|u,y}$ using the following proposal: given $a = (k, \theta)$ we sample $\tilde{k} \sim \sigma(k|\cdot)$ where $\sigma(k|\cdot)$ is a Markov kernel on $\mathbb{N}$, and then generate $\tilde{\theta}$ according to

   - If $\tilde{k} > k$ let $\tilde{\theta}_i = \theta_i$ for all $i = 1, \dots, k$ and then sample $\tilde{\theta}_{k+1}, \dots, \tilde{\theta}_{\tilde{k}}$ independently from the density τ_θ.
   - If $\tilde{k} \leq k$ let $\tilde{\theta}_i = \theta_i$ for all $i = 1, \dots, \tilde{k}$.

Remark 2.4.1. *We notice that the proposals described above are particular cases of the ones used in reversible jump Markov chain Monte Carlo [Green, 1995].*

For the Metropolis-Hastings algorithm associated to $\mathbb{L}_1^{u,y}(a|\cdot)$ the acceptance probability takes the form

$$\alpha_1(a, \tilde{a}) = \min\Big\{1, \exp\Big(-\Psi(u, \tilde{a}; y) + \Psi(u, a; y)\Big) \frac{\tau_{\tilde{\theta}}(\theta_k)}{\tau_\theta(\tilde{\theta}_k)}\Big\},$$

where recall $a = (k, \theta)$ and $\theta = (\theta_1, \dots, \theta_k)$ and $\tilde{a}$ is defined similarly.

For the Metropolis-Hastings algorithm associated to $\mathbb{L}_2^{u,y}(a|\cdot)$ the acceptance probability takes the form

$$\alpha_2(a,\tilde{a}) = \min\Big\{1, \frac{\sigma(k|\tilde{k})\nu_k(\tilde{k})}{\sigma(\tilde{k}|k)\nu_k(k)} \exp\Big(-\Psi(u,\tilde{a};y) + \Psi(u,a;y)\Big) H(k,\theta,\tilde{k},\tilde{\theta})\Big\},$$

where

$$H(k,\theta,\tilde{k},\tilde{\theta}) := \begin{cases} \prod_{i=1}^{k-\tilde{k}} \tau_{\tilde{\theta}}(\theta_{\tilde{k}+i}) & \text{if } k > \tilde{k}, \\ \Big(\prod_{i=1}^{\tilde{k}-k} \tau_{\theta}(\tilde{\theta}_{k+i})\Big)^{-1} & \text{if } \tilde{k} \geq k. \end{cases}$$

We notice that since each of the kernels $\mathbb{L}_1^{u,y}(a|\cdot)$ and $\mathbb{L}_2^{u,y}(a|\cdot)$ is defined by a Metropolis-Hastings algorithm, they leave the target $\nu_{a|u,y}$ invariant, and hence so does the kernel $\mathbb{L}^{u,y}(a|\cdot)$.

Remark 2.4.2. *If in the above the distribution τ_θ is, regardless of θ, the uniform distribution on the domain D, then the acceptance probabilities reduce, respectively, to*

$$\alpha_1(a,\tilde{a}) = \min\Big\{1, \exp\Big(-\Psi(u,\tilde{a};y) + \Psi(u,a;y)\Big)\Big\},$$

and

$$\alpha_2(a,\tilde{a}) = \min\left\{1, \frac{\sigma(k|\tilde{k})\nu_k(\tilde{k})}{\sigma(\tilde{k}|k)\nu_k(k)} \exp\Big(-\Psi(u,\tilde{a};y) + \Psi(u,a;y)\Big)\right\}.$$

Sampling Density-based Discretizations

Since in this case the dimension of θ is fixed, the calculation of the acceptance probabilities is straightforward and the details are omitted. We refer to Subsection 2.5.3 for a numerical example.

2.5 Numerical Examples

In this section we demonstrate the applicability of our framework and sampling approach in a variety of inverse problems. Our aim is illustrating the benefits and potential limitations of the methods; for this reason we consider inverse problems for which we have intuitive understanding of where the discretizations should concentrate, thus validating the performance of the proposed approach. Before discussing the numerical results, we summarize the main goals and outcomes of each set of experiments:

- In Subsection 2.5.1 we consider an inverse problem in mechanics [Bigoni et al., 2019], for which some observation settings highly influence the best choice of discretization while others inform it mildly. Our numerical results show that the gain afforded by grid learning is most clear whenever the observation locations highly influence the choice of discretization. We employ grid-based discretizations as described in Example 2.3.1 with an Euler discretization of the forward map. We also illustrate the applicability of the method in both finite and infinite-dimensional representations of the unknown parameter, showing a more dramatic effect in the latter.

- In Subsection 2.5.2 we consider an inverse problem in signal processing [Hairer et al., 2011], with a choice of observation locations that determine where the discretization should concentrate. Our numerical results show that the grids adapt to the expected region, and that the degrees of freedom in the discretization necessary to reconstruct the unknown is below that necessary to satisfy stability of the numerical method with uniform grids. We employ grid-based discretizations as described in Example 2.3.1 with an Euler-Maruyama discretization of the forward map.

- In Subsection 2.5.3 we consider an inverse problem in source detection, where the true hidden unknown determines how best to discretize the forward model. Our numerical results show that the grids adapt as expected. We employ density-based discretizations as described in Example 2.3.2 with a finite element discretization of the forward model.

2.5.1 Euler Discretization of ODEs: Estimation of the Young's Modulus of a Cantilever Beam

We consider an inhomogeneous cantilever beam clamped on one side ($x = 0$) and free on the other ($x = L$). Define $D = [0, L]$. Let $u(x)$ denote its Young's modulus and let $M(x)$ be a load applied onto the beam. Timoshenko's beam theory gives the displacement $z(x)$ of the beam and the angle of rotation $\varphi(x)$ through the coupled ordinary differential equations

$$\begin{cases} \frac{d}{dx}\left[\frac{u(x)}{2(1+r)}\Big(\varphi(x) - \frac{d}{dx}z(x)\Big)\right] = \frac{M(x)}{\kappa A}, \\ \frac{d}{dx}\Big(u(x)I\frac{d}{dx}\varphi(x)\Big) = \kappa A\frac{u(x)}{2(1+r)}\Big(\varphi(x) - \frac{d}{dx}z(x)\Big), \end{cases} \tag{2.5.1}$$

where r, κ, A, I are physical constants. Following [Bigoni et al., 2019], we consider the inverse problem of estimating the Young's modulus $u(x)$ from sparse observations of the displacement $z(x)$, where both u and z are functions from D to $\mathbb{R}$.

Let $\mathcal{F} : u \mapsto z$ be the solution map to equations (2.5.1). Let $\{s_i\}_{i=1}^m \subset D$ be the locations of the observation sensors, leading to the observation operator $\mathcal{O} : z \mapsto y \in \mathbb{R}^m$ defined coordinate-wise by

$$O_i(z) := \int_0^L z\varphi_i \mathrm{d}x, \qquad \varphi_i(x) := \frac{1}{\gamma_i}\exp\Big(-(s_i - x)^2/(2\delta^2)\Big), \qquad 1 \le i \le m,$$

where $\delta = 10^{-4}$ and γ_i is the normalizing constant such that $\int_0^L \varphi_i \mathrm{d}x = 1$. Data are generated according to the model

$$y = \mathcal{O} \circ \mathcal{F}(u) + \eta =: \mathcal{G}(u) + \eta,$$

where η denotes the observation error, which is assumed to follow a Gaussian distribution

$N(0, \gamma_{obs}^2 I)$. Notice that for system (2.5.1) with proper boundary conditions specified at $x = 0$, the displacement $z(x^\star)$ at any point $0 < x^\star < L$ depends only on the values $\{u(x) : x < x^\star\}$. Thus, we expect suitable discretizations of the forward model to refine finely only the region $\{0 < x < s_m\}$, where s_m is the right-most observation location. We will discuss this in detail in section 2.5.1.

Forward Discretization

To solve system (2.5.1) we employ a finite difference method. A family of numerical solutions can be parameterized by the set

$$\mathcal{A} := \Big\{ a = (k, \theta) : k \in \mathcal{K} \subset \{1, 2, \dots\}, \theta = [x_1, \dots, x_k] \in [0, L]^k \Big\},$$

where k is the number of grid points and θ are the grid locations. Precisely, for $a = (k, \theta) \in \mathcal{A}$, we first reorder θ so that

$$0 =: x_0 \leq x_1 \leq \cdots \leq x_k \leq x_{k+1} := L$$

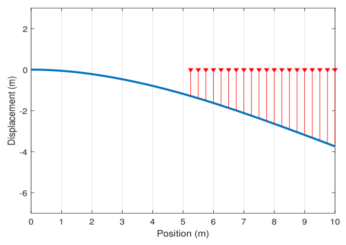
and we let

$$\mathcal{F}^a : u \mapsto z^a$$

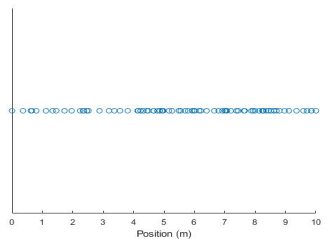
be the linearly interpolated explicit Euler finite difference solution to (2.5.1), discretized using the ordered grid θ. We also discretize the observation operator $\mathcal{O}$ using an Euler forward method, defined by

$$\mathcal{O}_i^a(z^a) = \sum_{j=0}^{k} z^a(x_j) \varphi_i(x_j)(x_{j+1} - x_j).$$

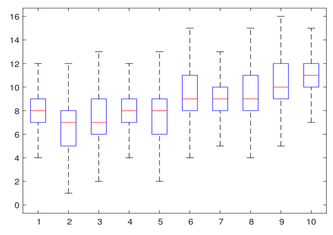
Finally $\mathcal{G}$ is approximated by $\mathcal{G}^a := \mathcal{O}^a \circ \mathcal{F}^a$.



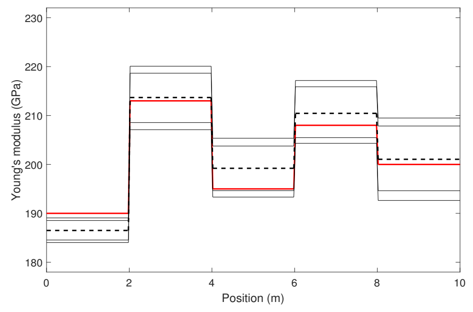
(a) Observation locations.



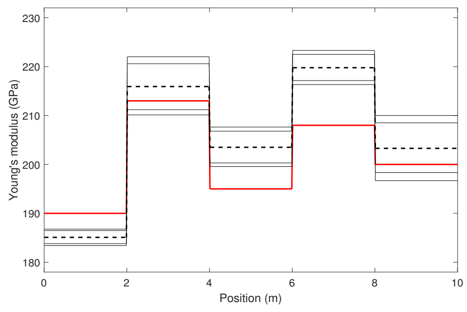
(b) A sampled grid.



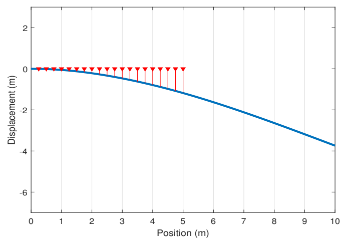
(c) Number of grid points.



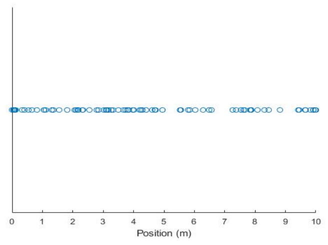
(d) True vs. posterior with grid learning.



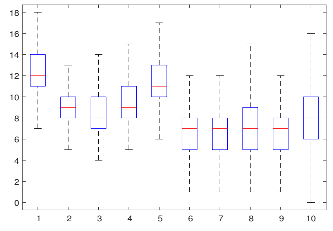
(e) True vs. posterior without grid learning.



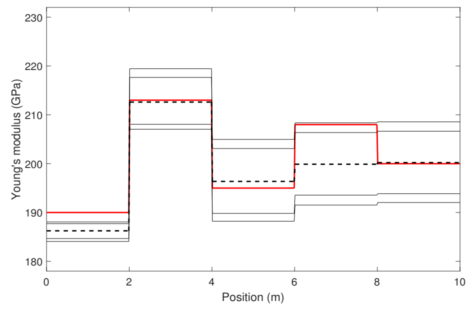
(f) Observation locations.



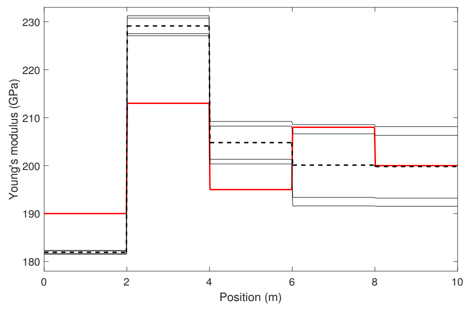
(g) A sampled grid.



(h) Number of grid points.

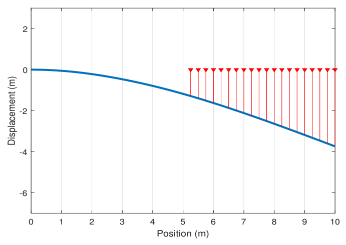


(i) True vs. posterior with grid learning.

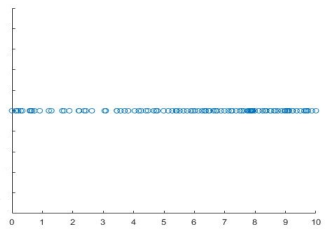


(j) True vs. posterior without grid learning.

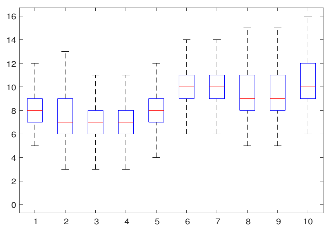
Figure 2.1: Reconstruction of a piece-wise constant Young's modulus. Two settings for the observation locations are considered, shown in Figures 2.1a, 2.1f. For each setting, Figures 2.1b and 2.1g show one sample from the marginal distribution $q_{a|y}(a)$ simulated by MCMC. Figures 2.1c and 2.1h report box-plots with the number of grid points that fall in each subinterval $[i-1, i]$, $i = 1, \dots, 10$. Figures 2.1d, 2.1i show the mean (dashed black) and the 5, 10, 90, 95-percentiles (thin black) of the marginal $q_{u|y}(u)$, versus the true value (red), with data-driven forward discretization. Figures 2.1e, 2.1j show the same results with a fixed uniform-grid discretization.



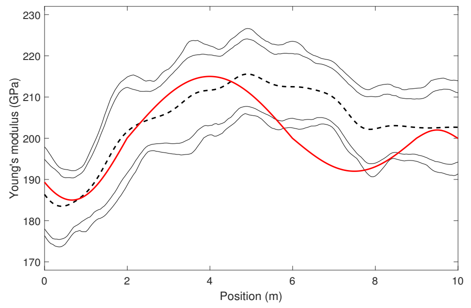
(a) Observation locations.



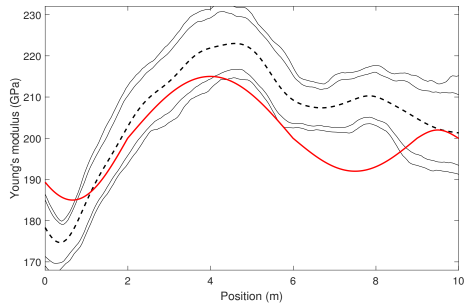
(b) A sampled grid.



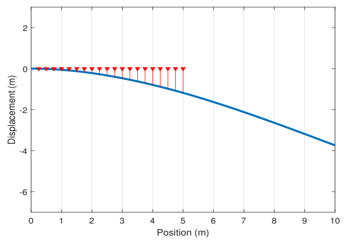
(c) Number of grid points.



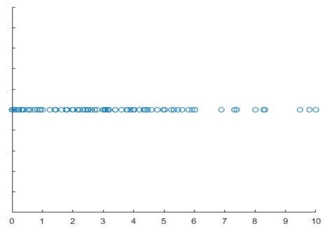
(d) True vs. posterior with grid learning.



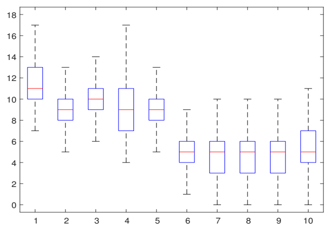
(e) True vs. posterior without grid learning.



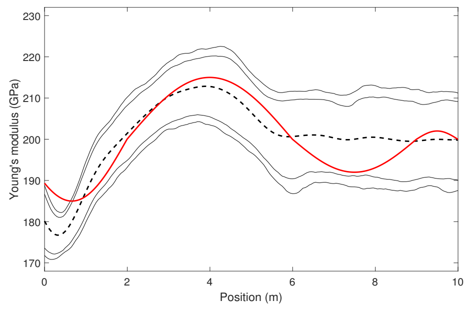
(f) Observation locations.



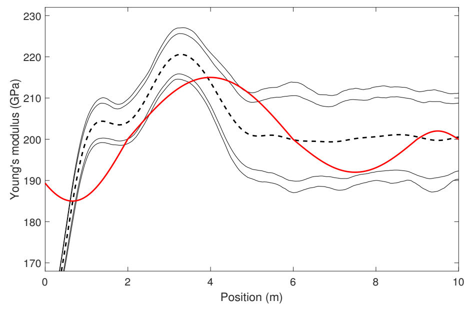
(g) A sampled grid.



(h) Number of grid points.



(i) True vs. posterior with grid learning.



(j) True vs. posterior without grid learning.

Figure 2.2: Reconstruction of a continuous Young's modulus. Two settings for the observation locations are considered, shown in Figures 2.2a, 2.2f. For each setting, Figures 2.2b and 2.2g show one sample from the marginal distribution $q_{a|y}(a)$ simulated by MCMC. Figures 2.2c and 2.2h report box plots with number of grid points that fall in each subinterval $[i-1, i]$, $i = 1, \ldots, 10$. Figures 2.2d, 2.2i show the mean (dashed black) and the 5, 10, 90, 95-percentiles (thin black) of the marginal $q_{u|y}(u)$, versus the true value (red), with data-driven forward discretization. Figures 2.2e, 2.2j show the same results with a fixed uniform-grid discretization.

Implementation Details and Numerical Results

For our numerical experiments we consider a beam of length $L = 10$ m, width $w = 0.1$ m and thickness $h = 0.3$ m. We use a Poisson ratio $r = 0.28$ and Timoshenko shear coefficient $\kappa = 5/6$. $A = wh$ represents the cross-sectional area of the beam and $I = wh^3/12$ is the second moment of inertia. We run a virtual experiment of applying a point mass of 5 kg at the end of the beam, as seen in blue in Figures 2.1a and 2.2a. We assume that the observations are gathered with error $\gamma_{obs}^2 = 10^{-3}$.

We first assume that the beam is made of 5 segments of different kinds of steel, each of length 2 m, with corresponding Young's moduli $u^* = \{u_i^*\}_{i=1}^5 = \{190, 213, 195, 208, 200 \text{ GPa}\}$. The prior on $u \in \mathsf{U} = \mathbb{R}^5$ is given by $p_u(u) = \mathcal{N}(u; 200\mathbf{1}, 25I_5)$ where $\mathbf{1}$ denotes the all-ones vector. For this case we assume that the number of grid points k is fixed to be $k = 85$, i.e., the prior $p_k(k)$ is a point mass. The grid locations θ are assumed to be a priori uniformly distributed in $[0, L]^k$. Results are reported in Figure 2.1. We next assume that the Young's modulus $u(x) \in \mathsf{U} = C([0, L]; \mathbb{R})$ varies continuously with x. We set a Gaussian process prior on u defined by $\mu_u = \mathcal{GP}(200, c)$ with $c(x, x') = 50 \exp\Big(-(x - x')^2/0.5\Big)$. For this case we assume that the prior on the number of grid points k follows a Poisson distribution with mean 60, i.e., $\nu_k(k) =$Poisson(60), and the grid locations θ still have a uniform prior given k.The true Young's modulus underlying the data and the reconstruction results are reported in Figure 2.2. Sampling is performed, both in the discrete and continuous settings, updating u and a alternately for a total number of $N = 1.2 \times 10^5$ iterations, with $\beta = 0.08$, $\zeta = 0.5$.

In both Figures 2.1 and 2.2 two settings are considered. In the first one observations are concentrated on the right side of the beam and in the second on the left. For reference, Figure 2.4 shows idealized posteriors considered, obtained with *very* fine discretizations $k = 500$, for each of the settings. Notice that for system (2.5.1) with proper boundary conditions specified at $x = 0$, the displacement $z(x_0)$ at any point $0 < x_0 < L$ depends only on the

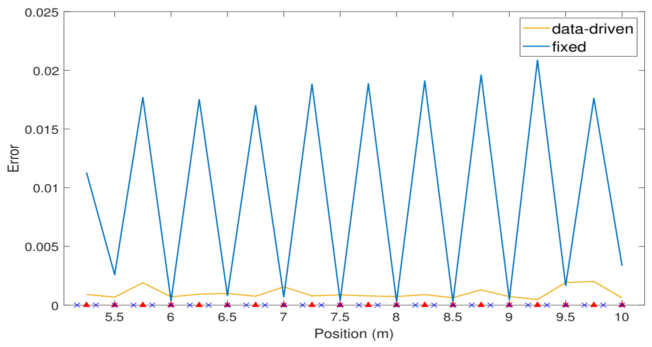
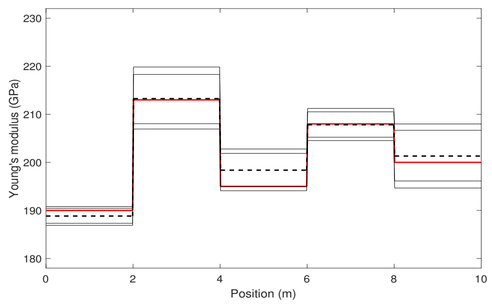


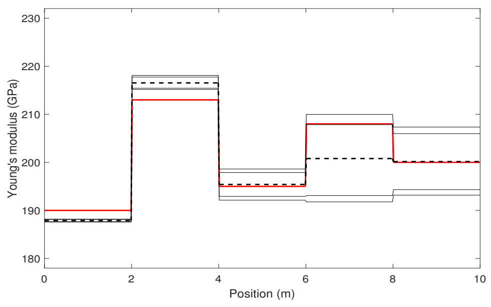
Figure 2.3: The reconstruction error with fixed-grid discretization (blue) and with data-driven grid discretization (orange). Red triangles are observations locations, while blue crosses are the grid points used in the fixed-grid discretization.

values $u(x)$ of Young's moduli with $x < x_0$. This implies that when observations are gathered on the left side of the beam, the posterior on $u(x)$ agrees with the prior on the right-side, and no resources should be on discretizing the forward map on that region. In that case our adaptive data-driven discretizations are strongly concentrated on the left, as shown in Figures 2.1g, 2.1h and 2.2g, 2.2h. However, when observations are gathered on the right side of the beam, the data is informative on $u(x)$ for all $0 < x < L$. In such case, Figures 2.1b, 2.1c and 2.2b, 2.2c show that the data-driven discretizations are concentrated on the right, but less heavily so. See Tables 2.7.1, 2.7.2 in the appendix for a more detailed description of the grid points distribution in both cases. Also, our results indicate that using data-driven discretizations will lead to a better estimation of the true Young's modulus, compared to fixed-grid discretizations. Additional results in the continuous Young's modulus setting are provided in the appendix. See Table 2.7.3 and Figure 2.8 for the averaged acceptence probability for u and a, and history of MCMC samples of the high-dimensional u at some fixed locations, indicating the stationarity of the Markov chain.

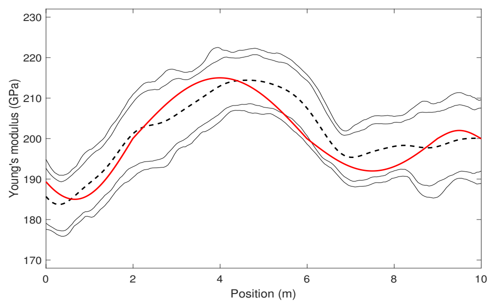
Let $(u^{(n)}, a^{(n)})$ be the output of the Gibbs sampling algorithm at iteration n. The



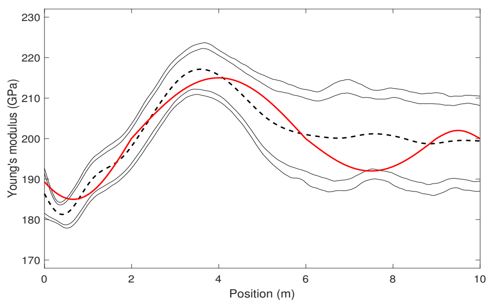
(a) Piece-wise constant, right observations.



(b) Piece-wise constant, left observations.



(c) Continuous, right observations.



(d) Continuous, left observations.

Figure 2.4: Idealized posterior $p_{u|y}(u)$, with mean (dashed black) and the 5, 10, 90, 95-percentiles (thin black), versus the true value (red).

reconstruction error is defined as follows:

$$e_r = \sqrt{\sum_{n=1}^{N} \left|\mathcal{G}^{a^{(n)}}(u^{(n)}) - \mathcal{G}(u)\right|^2} \ , \tag{2.5.2}$$

where $\mathcal{G}(u)$ is approximately calculated on a very fine grid. In Figure 2.3 we plot the reconstruction error for the second experiment where the Young's moduli is continuous and observations are gathered on the right-side. With fixed-grid discretization, the reconstruction error is small where the discretization matches the observation points. With adaptive data-driven discretizations the grid points will adaptively match the observation points in order to produce less error.

2.5.2 Euler-Maruyama Discretization of SDEs: a Signal Processing Application

Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be globally Lipschitz continuous and consider the SDE

$$dz(t) = f(z)\, dt + du, \quad 0 < t \leq T, \qquad z(0) = 0, \tag{2.5.3}$$

where u denotes d-dimensional Brownian motion. We aim to recover u from observations of the solution z. We suppose that the observations $y = [y_1, \ldots, y_m]$ are given by

$$y_i = z(t_i) + \eta_i, \qquad i = 1, \ldots, m, \tag{2.5.4}$$

where $\eta = [\eta_1, \ldots, \eta_m]$ is assumed to follow a centered Gaussian distribution with covariance Γ and

$$0 < t_1 < \cdots < t_m < T$$

are given observation times. Following [Hairer et al., 2011], we cast the problem in the setting of Section 2.2. First note that the solution to the integral equation

$$z(t) = \int_0^t f\big(z(s)\big)\, ds + u(t), \quad 0 \leq t \leq T, \tag{2.5.5}$$

defines a map

$$\mathcal{F} : C([0,T], \mathbb{R}^d) \to C([0,T], \mathbb{R}^d) \tag{2.5.6}$$

$$u \mapsto z\ . \tag{2.5.7}$$

Thus we set the input and output space to be $\mathsf{U} = \mathbb{Z} = C([0,T], \mathbb{R}^d)$.

Next we define an observation operator

$$\mathcal{O} : C([0,T], \mathbb{R}^d) \to \mathbb{R}^m$$

$$z \mapsto [z(s_1), \ldots, z(s_m)]$$

and set $\mathcal{G} = \mathcal{O} \circ \mathcal{F}$. We put as prior on u the standard d-dimensional Wiener measure, that we denote μ_u. Then the posterior distribution $\mu_{u|y}$ is given by Equation (2.2.4), which if $\Gamma = \gamma^2 I_m$ may be rewritten as

$$\frac{d\mu_{u|y}}{d\mu_u}(u) \propto \exp\left(\frac{1}{2\gamma^2}\sum_{i=1}^{m} \left|y_i - \mathcal{G}(u)(s_i)\right|^2\right). \tag{2.5.8}$$

Note that the likelihood does not involve evaluation of $\mathcal{F}(u)$ at times $t > s_m$, and hence changing the definition of $\mathcal{F}(u)(t)$ for $t > s_m$ does not change the posterior measure. Thus, we expect suitable discretizations of the forward model to refine finely only times up to the right-most observation.

Forward Discretization

For most nonlinear drifts f, the integral equation (2.5.6) cannot be solved in closed form and a numerical method needs to be employed. A family of numerical solutions may be parameterized by the set

$$\mathcal{A} := \Big\{a = (k, \theta) : \ k \in \mathsf{K} \subset \{1, 2, \ldots\},\ \theta = [t_1, \ldots, t_k] \in [0,T]^k\Big\}.$$

Precisely, for $a = (k, \theta) \in \mathcal{A}$ we define an approximate, Euler-Maruyama solution map

$$\mathcal{F}^a : C([0,T], \mathbb{R}^d) \to C([0,T], \mathbb{R}^d)$$

$$u \mapsto z^a$$

as follows. First, we reorder the t_j's so that

$$0 =: t_0 \leq t_1 \leq \ldots \leq t_k \leq t_{k+1} := T.$$

Then we define $z_j^a := z^a(t_j)$ as $z_0^a = 0$, and

$$z_{j+1}^a = z_j^a + (t_{j+1} - t_j) f(z_j^a) + u(t_{j+1}) - u(t_j), \quad 1 \leq j \leq k. \tag{2.5.9}$$

Finally, for $t \in (t_j, t_{j+1})$ we define $z^a(t)$ by linear interpolation of z_j^a and z_{j+1}^a.

Having defined the parameter space $\mathcal{A}$ we now describe a choice of prior distribution ν_a on $\mathcal{A}$ and the resulting combined prior $\nu_{u,a}$ on $(u, a) \in \mathsf{U} \times \mathcal{A}$. First we choose a prior ν_k on the number k of grid-points. Given the number of grid points k, assuming no knowledge on appropriate discretizations for the SDE (2.5.3) we put a uniform prior on grid locations $\theta = [t_1, \ldots, t_k]$.

Remark 2.5.1. *More information could be put into the prior. In particular it seems natural to impose that grids are finer at the beginning of the time interval.*

Implementation Details and Numerical Results

For our numerical experiments we considered the SDE (2.5.3) with $T = 10$ and double-well drift

$$f(t) = 10\, t\, (1 - t^2)/(1 + t^2). \tag{2.5.10}$$

We generated synthetic observation data y by solving (2.5.3) on a very fine grid, and then perturbing the solution at uniformly distributed times $t_i = 0.2i$, $i = 1, \ldots, 24 = m$ so that the last observation corresponds to time $t = 4.8$. The observation noise was taken to uncorrelated, $\Gamma = \gamma^2 I_m$, with $\gamma = 0.1$. The motivation for choosing this example is that there is certain intuition as to where one would desire the discretization grid-points to concentrate. Indeed, since all the observations t_i are in the interval $[0.2, 4.8]$ it is clear from Equation (2.5.8) that any discretization points $t_j \in (4.8, 10]$ will not contribute to better approximate $\mu_{u|y}$. In other words, those grid points would help in approximating $\mathcal{F}$ but not in approximating $\mathcal{G} = \mathcal{O} \circ \mathcal{F}$.

We report our results for a small grid-size $k = 24$. Similar but less dramatic effect was seen for larger grid size. Precisely, we chose our set of admissible grids to be given by

$$[t_0, \dots, t_{25}] : 0.01 = t_0 \leq t_1 \leq \dots \leq t_{25} = 10.$$

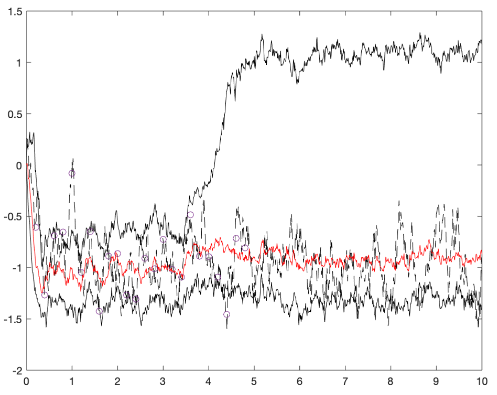
For implementation purposes, elements in the space $\mathsf{U} = C([0,T], \mathbb{R})$ were represented as vectors in $\mathbb{R}^{1000}$ containing their values on a uniform grid of step-size 0.1. We run these algorithms with parameter choices $N = 10^5$, $\beta = 0.1$, $\zeta = 0.5$.

The experiments show a successful reconstruction of the SDE path. Moreover, the grids concentrate in $[0, 4.8]$ in agreement with our intuition and the uncertainty quantification is satisfactory. In contrast, we see that when using the same number of grid points but on a uniform grid the Euler-Maruyama scheme is unstable, leading to a collapse of the MCMC algorithm. Then, the posterior constructed with a uniform grid completely fails at reconstructing the SDE path, and the uncertainty quantification is overoptimistic due to poor mixing of the chain.

2.5.3 Finite Element Discretization: Source Detection

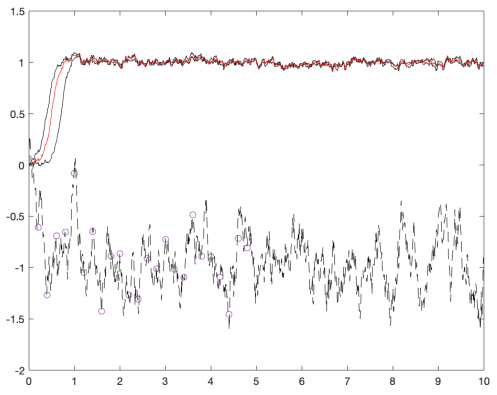
Consider the boundary value problem

$$\begin{cases} -\Delta z(x) = \delta(x-u), & x \in D, \\ z(x) = 0, & x \in \partial D, \end{cases} \tag{2.5.11}$$



(a) Data-driven discretization

.



(b) Uniform discretization.

Figure 2.5: Recovered SDE trajectory in the time-interval $t \in [0, 10]$. The true trajectory is shown in dashed black line. The posterior median is shown in red, and 5 and 95-percentiles are shown in black. The small circles denote the locations of the observations.

where $D = (0, 1) \times (0, 1) \subset \mathbb{R}^2$ is the unit square and δ is the Dirac function at the origin. We aim to recover the source location u from sparse observations

$$y_i = z(s_i) + \eta_i, \qquad i = 1, \ldots, m, \tag{2.5.12}$$

where $\eta = [\eta_1, \ldots, \eta_m]$ follows a centered Gaussian distribution with covariance Γ and $s_1, \ldots, s_m \in D \setminus \{u\}$ are observation locations. To cast the problem in the setting of Section 2.2, we let $\mathcal{F}$ be given by Green's function for the Laplacian on the unit square (which does not admit an analytical formula but can be computed e.g. via series expansions [Melnikov and Melnikov, 2006]) and $\mathcal{O}$ be defined by point-wise evaluation at the observation locations. The prior on u is the uniform distribution in the unit square D, which we denote $p_u(u)$. Since $\mathsf{U} = D$ is finite dimensional, the posterior has Lebesgue density given by Equation (2.2.3). We find this problem to be a good test, as there is a clear understanding that the data-driven mesh should concentrate around the source.

Forward Discretization

To solve equation (2.5.11) numerically we employ the finite element method. The use of uniform grid is here wasteful, as the mesh should ideally concentrate around the unknown source u.

We will use grids obtained as the Delauny triangulation of central Voronoi tessellations $\{V_i\}_{i=1}^k$ and generators $\{x_i\}_{i=1}^k$, where each $x_i \in D$ and $V_i \subset D$. This can be calculated as the solution of the optimization problem, parameterized by a probability density ρ on D:

$$\min_{\{x_i\}\subset D, \{V_i\}} \sum_{i=1}^{k} \int_{V_i} \rho(x) \|x - x_i\|^2 \mathrm{d}x \tag{2.5.13}$$

subject to the constraint that $\{V_i\}_{i=1}^k$ is a tessellation of D. One can refer to [Du and Gunzburger, 2002] for more details. For a fixed density ρ and integer k we denote the optimal grid points by $\{x_{\rho,i}\}_{i=1}^k$. Then the approximated solution map is defined as

$$\mathcal{F}^a : D \to H_0^1(D) \tag{2.5.14}$$

$$u \mapsto z^a \tag{2.5.15}$$

where z^a is given by the finite element solution of equation (2.5.11) with respect to (the Delauny triangulation of) the grid points $\{x_{\rho,i}\}_{i=1}^k$. Details on the creation of grid for prescribed parameters ρ and k will be discussed below.

In the spirit of adapting the grid to favor the ones maximizing the model evidence, we constrain ρ to belong to a family of parametric densities $\Pi = \{\rho(x;\theta) | \theta \in \mathbb{R}^P\}$ where $\rho(x;\theta) = \mathrm{Beta}(\alpha_1, \beta_1) \times \mathrm{Beta}(\alpha_2, \beta_2)$ is the product measure of two Beta distributions. Therefore in this case $\theta = (\alpha_1, \beta_1, \alpha_2, \beta_2)$ and $P = 4$. Each pair (k, θ) describes a member in the discretization family $\mathcal{A}$, where k controls the number of grid points, while θ controls how these grid points are distributed in the spatial dimension.

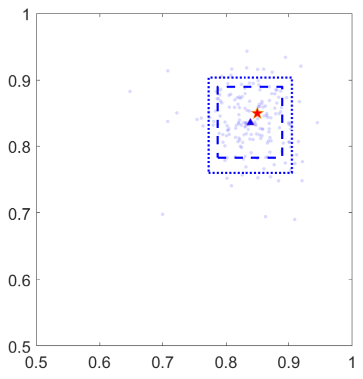
Implementation Details and Numerical Results

We solved equation (2.5.11) on a fine grid $k = 2000$ with the true point source $u^* = (0.85, 0.85)$. The observation locations were $\{s_1, \dots, s_{25}\} = \{0.5, 0.6, 0.7, 0.8, 0.9\} \times \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Observation noise was uncorrelated with $\Gamma = \gamma^2 I_{25}$, $\gamma = 0.05$.

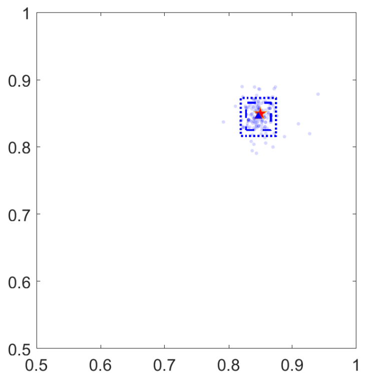
In this example the prior of u is the uniform distribution on $D = (0,1) \times (0,1)$ and u is initialized at $(0.2, 0.2)$. The parameters θ are also set to have a uniform prior $\theta = (\alpha_1, \beta_1, \alpha_2, \beta_2) \sim \mathit{Uniform}\big([1, 10]^4\big)$. We initialize $(\alpha_1, \beta_1, \alpha_2, \beta_2) = (1, 1, 1, 1)$, which corresponds to (near) uniform grid in D. For simplicity, in this experiment we set a point mass prior on k, with $k = 100$. We run the algorithm with $N = 10^4$.

We compare our algorithm to the traditional method where we fix a uniform grid in D and run the MCMC algorithm only on u. We found out that with the same number of grid points, our data-driven approach gives a posterior distribution $q_{u|y}$ that is more concentrated around the true location of the point source, as shown in Figure 2.6a and 2.6b. Also, Figure 2.6c shows that the adaptive discretization is concentrated at the top right corner of the region, where the hidden point source u^* is located.

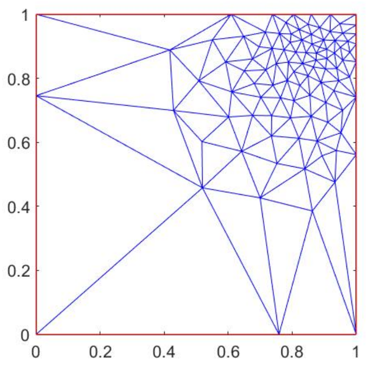
Next we show that data-driven discretizations of the forward map can be employed to provide improved uncertainty quantification of the PDE solution, and not only to better reconstruct the unknown input. To illustrate this, we approximate the pushforward distribution $\mathcal{F}_\sharp(q_{u|y})$ in three different ways, as shown in Figure 2.7. Let $(u^{(n)}, a^{(n)})$ denote the output of the Gibbs sampling algorithm at iteration n. We first consider the traditional method where the grid is fixed and uniform, that is, $a^{(n)} = a$ is fixed. Then the pushforward distribution can be well-approximated by $\{\mathcal{F}^a(u^{(n)})\}_{n=1}^N$, for N large enough. We then consider the same setting except that $\mathcal{F}^a$ is replaced by a forward map $\mathcal{F}$ computed in a fine grid $k = 2000$ and the pushforward is approximated by $\{\mathcal{F}(u^{(n)})\}_{n=1}^N$. Finally we consider a data-driven setting stemming from our algorithm, where the pushforward distribution is approximated by $\{\mathcal{F}^{a^{(n)}}(u^{(n)})\}_{n=1}^N$. We see that our algorithm reconstructs well the solution



(a) Fixed grid.



(b) Data-driven grid.



(c) Grid generated in the last iteration.

Figure 2.6: Figures 2.6a and 2.6b show the posterior distribution $q(u|y)$, where the grid is fixed and uniform in 2.6a, and data-driven in 2.6b. Red star indicates the true location of the source, blue dots are random samples from the posterior, blue triangle is the posterior mean, and dash (resp. dotted) lines correspond to the 90% (resp. 95%) coordinate-wise credible regions. Figure 2.6c shows the grid generated in the last iteration of the MCMC update.

to the PDE, with a more accurate mean and a smaller variance.

2.6 Conclusions and Open Directions

- We have shown that, in a variety of inverse problems, the observations contain useful information to guide the discretization of the forward model, allowing a better reconstruction of the unknown than using uniform grids with the same number of degrees of freedom. Despite these results being promising, it is important to note that updating the discretization parameters may be costly in itself, and may result in slower mixing of the MCMC methods. For this reason, we envision that the proposed approach may have more potential when the computational solution of the inverse problem is very sensitive to the discretization of the forward map and discretizing it is expensive. We also believe that density-based discretizations may help in alleviating the cost of discretization learning.

- An interesting avenue of research stemming from this work is the development of prior

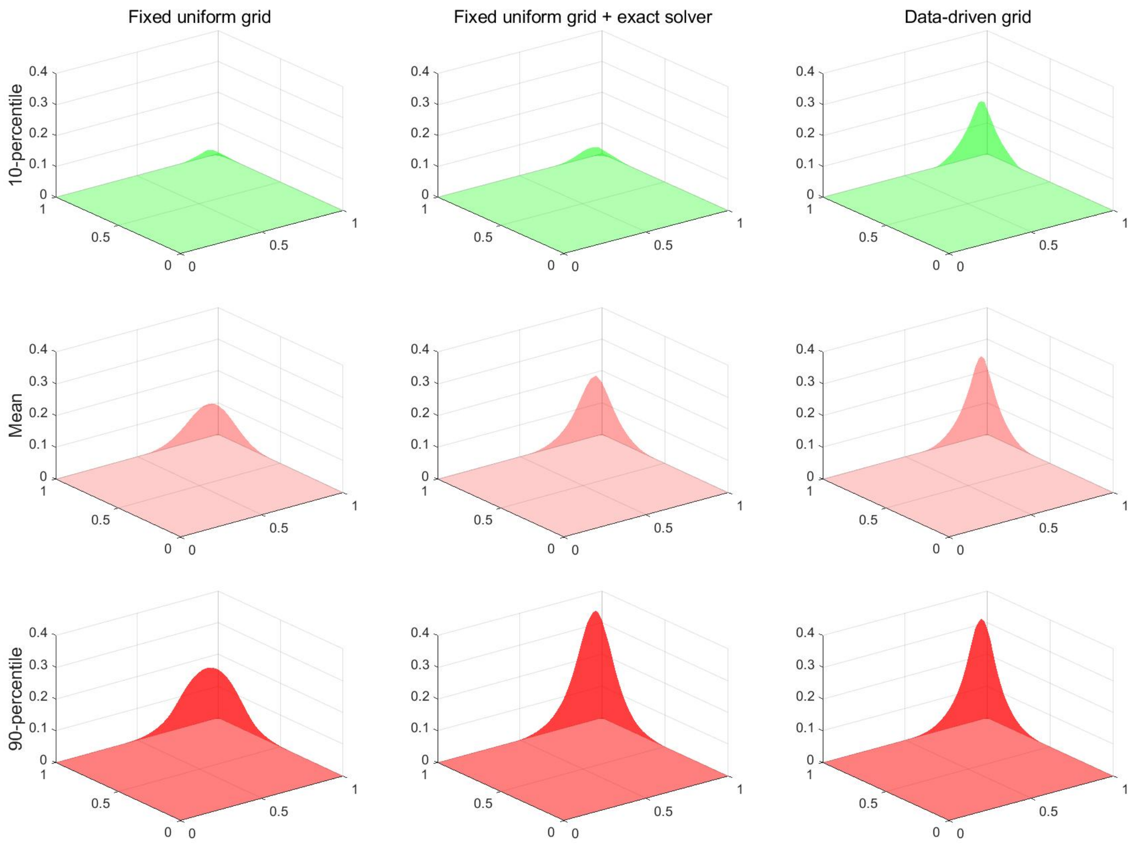


Figure 2.7: The mean, 10 and 90-percentile of the pushforward distribution $\mathcal{F}_\sharp(q_{u|y})$ under three different settings: (1) Both the posterior $q_{u|y}$ and its pushforward $\mathcal{F}_\sharp(q_{u|y})$ are computed on a fixed and uniform grid; (2) The posterior $q_{u|y}$ is computed on a fixed and uniform grid, and its pushforward $\mathcal{F}_\sharp(q_{u|y})$ is calculated using a (nearly) exact solver; (3) Both the posterior and its pushforward are computed on a data-driven grid.

discretization models that are informed by numerical analysis of the forward map $\mathcal{F}$, while recognizing the uncertainty in the best discretization of the forward model $\mathcal{G}$. Moreover, more sophisticated prior models beyond the product structure considered here should be investigated.

- Topics for further research include the development of new local proposals and sampling algorithms for grid-based discretizations, and the numerical implementation of the approach in computationally demanding inverse problems beyond the proof-of-concept ones considered here.

2.7 Appendix

2.7.1 Algorithm Pseudo-Code

Algorithm 2.7.1 Metropolis within-Gibbs

```
Input parameters: $\beta$ (pCN step-size), $\zeta$ (probability of location moves), $N$ (sample size).
Choose $(u^{(1)}, a^{(1)}) \in \mathsf{U} \times \mathcal{A}$.
for $n = 1 : N$ do
    Stage I Do a pCN move to update $u$ given $a, y$ :
    i) Propose $\tilde{u}^{(n)} = \sqrt{1-\beta^2}\, u^{(n)} + \beta v^{(n)}, \qquad v^{(n)} \sim \mu_u$.
    ii) Set $u^{(n+1)} = \tilde{u}^{(n)}$ with probability
        $$a(u^{(n)}, \tilde{u}^{(n)}) = \min\left\{1, \exp\Big(\Psi(u^{(n)}, a^{(n)}; y) - \Psi(\tilde{u}^{(n)}, a^{(n)}; y)\Big)\right\}.$$
    iii) Set $u^{(n+1)} = u^{(n)}$ otherwise.
    Stage II Update $a = (k, \theta)$ given $u$ and $y$.
    Stage IIa With probability $\zeta$, update $\theta$ given $u, y$ with a grid re-location step:
    i) Propose $\tilde{a}^{(n)}$ by picking one of the $k$ interior grid points of $a^{(n)}$ uniformly at random,
       and replacing it by a uniform draw in $D$.
    ii) Set $a^{(n+1)} = \tilde{a}^{(n)}$ with probability
        $$\alpha(a^{(n)}, \tilde{a}^{(n)}) = \min\left\{1, \exp\Big(\Psi(u^{(n+1)}, a^{(n)}; y) - \Psi(u^{(n+1)}, \tilde{a}^{(n)}; y)\Big)\right\}.$$
    iii) Set $a^{(n+1)} = a^{(n)}$ otherwise.
    Stage IIb Otherwise, (with probability $1-\zeta$) update $k$ with a birth/death step:
    i) Propose a new number $\tilde{k}^{(n)}$ of grid-points.
    ii) If $\tilde{k}^{(n)} \le k^{(n)}$ remove uniformly chosen grid-points.
    iii) If $\tilde{k}^{(n)} > k^{(n)}$ draw required number of new grid points uniformly at random in $D$.
    iv) Set $a^{(n+1)} = \tilde{a}^{(n)}$ with probability
        $$\alpha(a^{(n)}, \tilde{a}^{(n)}) = \min\left\{1, \frac{\nu_k(\tilde{k}^{(n)})}{\nu_k(k^{(n)})} \exp\Big(\Psi(u^{(n+1)}, a^{(n)}; y) - \Psi(u^{(n+1)}, \tilde{a}^{(n)}; y)\Big)\right\}.$$
    v) Set $a^{(n+1)} = a^{(n)}$ otherwise.
end for
```

2.7.2 Additional Results for Section 2.5.1

	(0, 1)	(1, 2)	(2, 3)	(3, 4)	(4, 5)	(5, 6)	(6, 7)	(7, 8)	(8, 9)	(9, 10)
2-3	0.0015	0.0047	0.0116	0.0094	0.0164	0	0	0	0	0
4-5	0.0791	0.1089	0.2003	0.1721	0.1875	0.0260	0.0967	0.0363	0.0554	0
6-7	0.3345	**0.3940**	**0.4002**	**0.4017**	**0.3956**	0.2309	0.2817	0.1789	0.3529	0.0578
8-9	**0.3956**	0.3412	0.2785	0.2967	0.2907	**0.3548**	**0.3011**	**0.4835**	**0.3944**	0.3808
10-11	0.1548	0.1268	0.0920	0.0983	0.0924	0.2579	0.2612	0.2565	0.1635	**0.3886**
12-13	0.0289	0.0230	0.0153	0.0199	0.0159	0.0988	0.0550	0.0425	0.0312	0.1524
14-15	0.0052	0.0012	0.0016	0.0019	0.0016	0.0255	0.0044	0.0020	0.0027	0.0195
16-17	0.0004	0.0003	0.0004	0	0	0.0051	0	0.0002	0	0.0009
18-19	0	0	0	0	0	0.0009	0	0	0	0

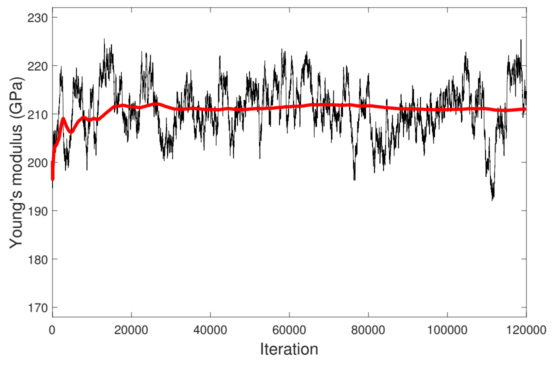
Table 2.7.1: Distribution of grid points when observations are concentrated on the right, in the piecewise-constant Young' modulus case. Element on i^{th} row and j^{th} column represents the posterior probability of having i grid points in the subinterval j.

	(0, 1)	(1, 2)	(2, 3)	(3, 4)	(4, 5)	(5, 6)	(6, 7)	(7, 8)	(8, 9)	(9, 10)
2-3	0	0	0	0	0	0.1712	0.1481	0.1435	0.1259	0.0605
4-5	0	0.0496	0.0076	0	0.0325	**0.3420**	**0.3139**	**0.3183**	**0.3162**	0.2362
6-7	0	0.2488	0.1263	0.0257	0.2238	0.2860	0.2995	0.3133	0.3087	**0.3360**
8-9	0.0497	**0.3785**	0.3236	0.2174	**0.4048**	0.1341	0.1542	0.1508	0.1615	0.2316
10-11	0.2055	0.2390	**0.3357**	**0.4114**	0.2623	0.0383	0.0540	0.0483	0.0554	0.0954
12-13	**0.3384**	0.0717	0.1573	0.2492	0.0705	0.0062	0.0118	0.0097	0.0131	0.0274
14-15	0.2516	0.0111	0.0424	0.0789	0.0061	0.0012	0.0016	0.0009	0.0017	0.0053
16-17	0.1163	0.0013	0.0070	0.0155	0	0	0	0	0	0.0009
18-19	0.0321	0	0.0009	0.0018	0	0	0	0	0	0
20-21	0.0053	0	0	0	0	0	0	0	0	0
22-23	0.0007	0	0	0	0	0	0	0	0	0

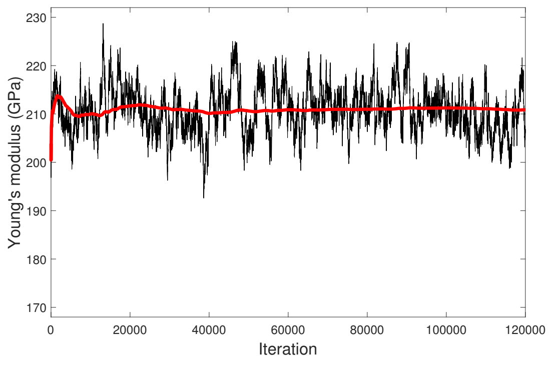
Table 2.7.2: Distribution of grid points when observations are concentrated on the left, in the piecewise-constant Young' modulus case. Element on i^{th} row and j^{th} column represents the posterior probability of having i grid points in the subinterval j.

	right obs.	left obs.
u	0.2728	0.4590
a	0.1953	0.2314

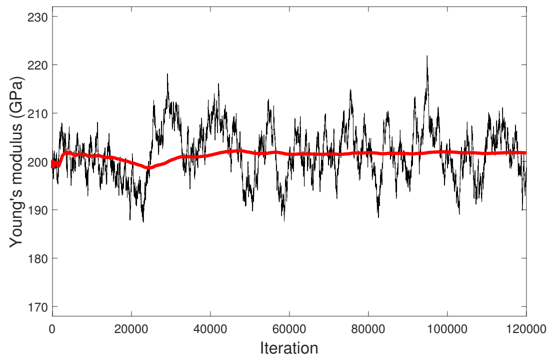
Table 2.7.3: Averaged acceptance probability of u and a respectively, in the continuous Young' modulus case.



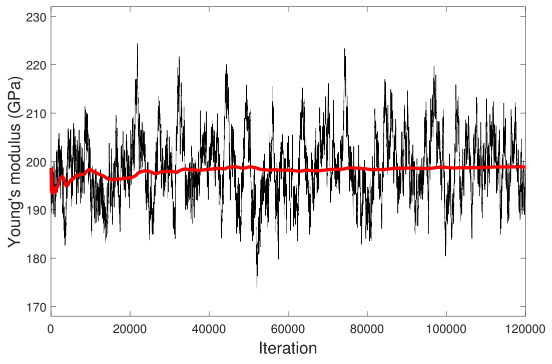
(a) Continuous, right observations, $x = 4$.



(b) Continuous, left observations, $x = 4$.



(c) Continuous, right observations, $x = 8$.



(d) Continuous, left observations, $x = 8$.

Figure 2.8: History of MCMC samples (black line) and running sample averages (red line) of continuous Young's modulus $u(x)$, at fixed locations $x = 4$ and $x = 8$ repectively, suggesting stationarity of the Markov chain.

CHAPTER 3

AUTODIFFERENTIABLE ENSEMBLE KALMAN FILTERS

3.1 Introduction

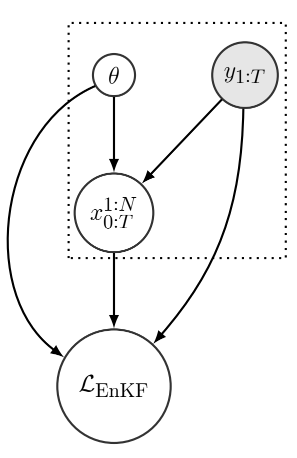
Time series of data arising across geophysical sciences, remote sensing, automatic control, and a variety of other scientific and engineering applications often reflect observations of an underlying dynamical system operating in a latent state-space. Estimating the evolution of this latent state from data is the central challenge of data assimilation (DA) [Jazwinski, 2007; Evensen, 2009; Sanz-Alonso et al., 2018; Law et al., 2015; Reich and Cotter, 2015]. However, in these and other applications, we often lack an accurate model of the underlying dynamics, and the dynamical model needs to be learned from the observations to perform DA. This paper introduces auto-differentiable ensemble Kalman filters (AD-EnKFs), a machine learning (ML) framework for the principled co-learning of states and dynamics. This framework enables learning in three core categories of unknown dynamics: (a) parametric dynamical models with unknown parameter values; (b) fully-unknown dynamics captured using neural network (NN) surrogate models; and (c) inaccurate or partially-known dynamical models that can be improved using NN corrections. AD-EnKFs are designed to scale to high-dimensional states, observations, and NN surrogate models.

In order to describe the main idea behind the AD-EnKF framework, let us introduce briefly the problem of interest. Our setting will be formalized in §3.2 below. Let $x_{0:T} := \{x_t\}_{t=0}^{T}$ be a time-homogeneous *state process* with transition kernel $p_\theta(x_t|x_{t-1})$ parameterized by a vector θ. For instance, θ may contain unknown parameters of a parametric dynamical model or the parameters of a NN surrogate model for the dynamics. Our aim is to learn θ from partial and noisy observations $y_{1:T} := \{y_t\}_{t=1}^{T}$ of the state, and thereby learn the unknown dynamics and estimate the state process. The AD-EnKF framework learns θ iteratively. Each iteration consists of three steps: (i) use EnKF to compute an estimate

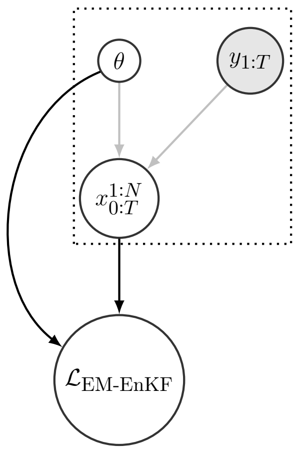
$\mathcal{L}_{\text{EnKF}}(\theta)$ of the data log-likelihood $\mathcal{L}(\theta) := \log p_\theta(y_{1:T})$; (ii) use auto-differentiation ("autodiff") to compute the gradient $\nabla_\theta \mathcal{L}_{\text{EnKF}}(\theta)$; and (iii) take a gradient ascent step. Filtered estimates of the state are obtained using the learned dynamics.

The EnKF, reviewed in §3.3, estimates the data log-likelihood using an ensemble of particles. Precisely, given a transition kernel $p_\theta(x_t|x_{t-1})$, the EnKF generates particles $x_{0:T}^{1:N} := \{x_t^n\}_{\substack{t=0,\dots T\\ n=1,\dots,N}}$; here x_t^n represents a generic particle that approximates the state x_t at discrete time $t \in \{0, \dots, T\}$, and N denotes the ensemble size. The log-likelihood estimate $\mathcal{L}_{\text{EnKF}}(\theta)$ depends on θ through these particles and also through the given transition kernel. Differentiating the map $\theta \mapsto \mathcal{L}_{\text{EnKF}}(\theta)$ in step (ii) of AD-EnKF involves differentiating *both* the map $\theta \mapsto x_{0:T}^{1:N}$ from parameter to EnKF particles and the map $(\theta, x_{0:T}^{1:N}) \mapsto \mathcal{L}_{\text{EnKF}}(\theta)$ from parameters and EnKF particles to EnKF log-likelihood estimate. *A key feature of our approach is that $\theta \mapsto \mathcal{L}_{EnKF}(\theta)$ can be auto-differentiated using the reparameterization trick ([Kingma and Welling, 2014] and §3.4.1) and autodiff capabilities of NN software libraries* such as PyTorch [Paszke et al., 2019], JAX [Bradbury et al., 2018], and Tensorflow [Abadi et al., 2016]. Automatic differentiation is different from numerical differentiation in that derivatives are computed exactly through compositions of elementary functions whose derivatives are known, as opposed to finite difference approximations that cause discretization errors.

The AD-EnKF framework represents a significant conceptual and methodological departure from existing approaches to blend DA and ML based on the expectation-maximization (EM) framework, see Fig. 3.1 below. Specifically, at each iteration, EM methods that build on the EnKF [Pulido et al., 2018; Brajard et al., 2020; Bocquet et al., 2020] employ a surrogate likelihood $\mathcal{L}_{\text{EM-EnKF}}\big(\theta\,; x_{0:T}^{1:N}\big)$ where the particles $x_{0:T}^{1:N}$ are generated by EnKF and *fixed.* Importantly, EM methods compute gradients used to learn dynamics by differentiating only through the θ-dependence in $\mathcal{L}_{\text{EM-EnKF}}$ that does not involve the particles. In particular, in contrast to AD-EnKF, the map $\theta \mapsto x_{0:T}^{1:N}$ from parameter to EnKF particles is *not* differentiated. Moreover, the performance of EM methods is sensitive to the specific choice of EnKF algorithm in use, and the tuning of algorithmic parameters of EM can be challenging [Brajard et al., 2020; Bocquet et al., 2020]. Our numerical experiments suggest that, even when optimally tuned, EM methods underperform AD-EnKF in high-dimensional regimes. The better performance of AD-EnKF may be explained by the additional gradient information obtained by differentiating the map $\theta \mapsto x_{0:T}^{1:N}$.



(a) AD-EnKF



(b) EM-EnKF

With AD-EnKF, parameters θ and observations $y_{1:T}$ are used to generate EnKF particles $x_{0:T}^{1:N}$; the particles together with θ and $y_{1:T}$ are used to compute the likelihood $\mathcal{L}_{\text{EnKF}}$, and the gradient $\nabla_\theta \mathcal{L}_{\text{EnKF}}$ explicitly accounts for the map from θ to the particles $x_{0:T}^{1:N}$. In contrast, with EM-EnKF, the likelihood $\mathcal{L}_{\text{EM-EnKF}}$ is a function of θ and *fixed* particles $x_{0:T}^{1:N}$ generated by EnKF, so that computing the gradient $\nabla_\theta \mathcal{L}_{\text{EM-EnKF}}$ does *not* account for the map from θ to the particles $x_{0:T}^{1:N}$.

Figure 3.1: Computational graph of AD-EnKF and EM-EnKF. Dashed squares represent computations performed by the EnKF. Gray arrows in (b) indicate that the construction of $\mathcal{L}_{\text{EM-EnKF}}$ is performed in two steps: (1) obtain $x_{0:T}^{1:N}$ from θ and $y_{1:T}$ (gray arrows); and (2) use θ and $x_{0:T}^{1:N}$ (no longer seen as a function of θ) to define $\mathcal{L}_{\text{EM-EnKF}}$. In contrast, those lines are black in (a), indicating that in AD-EnKF the particles $x_{0:T}^{1:N}$ in $\mathcal{L}_{\text{EnKF}}$ are seen as varying with θ.

The AD-EnKF framework also represents a methodological shift from existing differentiable particle filters [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018]. Similar to AD-EnKF, these methods rely on autodiff of a map $\theta \mapsto \mathcal{L}_{\text{PF}}(\theta)$, where the log-likelihood estimate $\mathcal{L}_{\text{PF}}(\theta)$ depends on θ through weighted particles $(w_{0:T}^{1:N}, x_{0:T}^{1:N})$ obtained by running a particle filter (PF) with transition kernel $p_\theta(x_t|x_{t-1})$. However, the use of PF suffers from two caveats. First, it is not possible to auto-differentiate directly through the PF resampling steps [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018]. Second, while the PF log-likelihood estimates are consistent, their variance can be large, especially in high-dimensional systems. Moreover, their *gradient*, which is the quantity used to perform gradient ascent to learn θ, is not consistent [Corenflos et al., 2021].

3.1.1 Contributions

This paper seeks to set the foundations and illustrate the capabilities of the AD-EnKF framework through rigorous theory and systematic numerical experiments. Our main contributions are:

- We develop new theoretical convergence guarantees for the large sample EnKF estimation of log-likelihood gradients in linear-Gaussian settings (Theorem 3.3.2).
- We combine ideas from online training of recurrent networks (specifically, Truncated Backpropagation Through Time – TBPTT) with the learning of AD-EnKF when the data sequence is long, i.e. T is large.
- We provide numerical evidence of the superior estimation accuracies of log-likelihoods and gradients afforded by EnKF relative to PF methods in high-dimensional settings. In particular, we illustrate the importance of using localization techniques, developed in the DA literature, for EnKF log-likelihood and gradient estimation, and the corresponding performance boost within AD-EnKF.
- We conduct a numerical case study of AD-EnKF on the Lorenz-96 model [Lorenz, 1996], considering parameterized dynamics, fully-unknown dynamics, and correction of an inaccurate model. The importance of the Lorenz-96 model in geophysical applications and for testing the efficacy of filtering algorithms is highlighted, for instance, in [Majda and Harlim, 2012; Law and Stuart, 2012; Law et al., 2016a; Brajard et al., 2020]. Our results show that AD-EnKF outperforms existing methods based on EM or differentiable PFs. The improvements are most significant in challenging high-dimensional and partially-observed settings.

3.1.2 Related Work

The EnKF algorithm was developed as a state estimation tool for DA [Evensen, 1994] and is now widely used in numerical weather prediction and geophysical applications [Szunyogh et al., 2008; Whitaker et al., 2008]. Recent reviews include [Houtekamer and Zhang, 2016; Katzfuss et al., 2016; Roth et al., 2017]. The idea behind the EnKF is to propagate N equally-weighted particles through the dynamics and assimilate new observations using Kalman-type updates computed with empirical moments. When the state dimension d_x is high and the ensemble size N is moderate, traditional Kalman-type methods require $O(d_x^2)$ memory to store full covariance matrices, while storing empirical covariances in EnKFs only requires $O(Nd_x)$ memory. The use of EnKF for joint learning of state and model parameters by *state augmentation* was introduced in [Anderson, 2001], where EnKF is run on an augmented state-space that includes the state and parameters. However, this approach requires one to design a pseudo-dynamic for the parameters which needs careful tuning and can be problematic when certain types of parameters (e.g., error covariance matrices) are involved [Stroud and Bengtsson, 2007; DelSole and Yang, 2010] or if the dimension of the parameters is high. In this paper, we employ EnKFs to approximate the data log-likelihood. The use of EnKF to perform derivative-free maximum likelihood estimation (MLE) is studied in [Stroud et al., 2010; Pulido et al., 2018]. An empirical comparison of the likelihood computed using the EnKF and other filtering algorithms is made in [Carrassi et al., 2017]; see also [Hannart et al., 2016; Metref et al., 2019]. The paper [Drovandi et al., 2021] uses EnKF likelihood estimates to design a pseudo-marginal Markov chain Monte Carlo (MCMC) method for Bayesian inference of model parameters. The works [Stroud and Bengtsson, 2007; Stroud et al., 2018] propose online Bayesian parameter estimation using the likelihood computed from the EnKF under a certain family of conjugate distributions. However, to the best of our knowledge, there is no prior work on state and parameter estimation that utilizes gradient information of the EnKF likelihood.

The embedding of EnKF and ensemble Kalman smoothers (EnKS) into the EM algorithm for MLE [Dempster et al., 1977; Bishop, 2006] has been studied in [Tandeo et al., 2015; Ueno and Nakamura, 2014; Dreano et al., 2017; Pulido et al., 2018], with a special focus on estimation of error covariance matrices. The expectation step (E-step) is approximated with EnKS under the Monte Carlo EM framework [Wei and Tanner, 1990]. In addition, [Brajard et al., 2020; Nguyen et al., 2019] incorporate deep learning techniques in the maximization step (M-step) to train NN surrogate models. The paper [Bocquet et al., 2020] proposes Bayesian estimation of model error statistics, in addition to an NN emulator for the dynamics. On the other hand, [Ueno and Nakamura, 2016; Cocucci et al., 2021] consider online EM methods for error covariance estimation with EnKF. Although gradient information is used during the M-step to train the surrogate model [Brajard et al., 2020; Nguyen et al., 2019; Bocquet et al., 2020], these methods do not auto-differentiate through the EnKF (see Fig. 3.1), and accurate approximation of the E-step is hard to achieve with EnKF or EnKS.

Another popular approach for state and parameter estimation are particle filters (PFs) [Gordon et al., 1993; Doucet and Johansen, 2009] that approximate the filtering step by propagating samples with a kernel, reweighing them with importance sampling, and resampling to avoid weight degeneracy. PFs give an unbiased estimate of the data likelihood [Del Moral, 2004; Andrieu et al., 2010]. Based upon this likelihood estimate, a particle MCMC Bayesian parameter estimation method is designed in [Andrieu et al., 2010]. Although PF likelihood estimates are unbiased, they suffer from two important caveats. First, their variance can be large, as they inherit the weight degeneracy of importance sampling in high dimensions [Snyder et al., 2008; Bocquet et al., 2010; Agapiou et al., 2017; Sanz-Alonso, 2018; Sanz-Alonso and Wang, 2021]. Second, while the propagation and reweighing steps of PFs can be auto-differentiated, the resampling steps involve discrete distributions that cannot be handled by the reparameterization trick. For this reason, previous differentiable PFs omit autodiff of the resampling step [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018], introducing a bias. To address this issue, the resampling step can be replaced with a differentiable optimal transport map [Corenflos et al., 2021], but construction of this map can be computationally expensive.

An alternative to MLE methods is to optimize a lower bound of the data log-likelihood with variational inference (VI) [Bishop, 2006; Kingma and Welling, 2014; Ranganath et al., 2014]. The posterior distribution over the latent states is approximated with a parametric distribution and is jointly optimized with model parameters defining the underlying state-space model. In this direction, variational sequential Monte Carlo (VSMC) methods [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018] construct the lower bound using a PF algorithm. Moreover, the proposal distribution of the PF is parameterized and jointly optimized with model parameters defining the state-space model. Although VSMC methods provide consistent data log-likelihood estimates, they suffer from the same two caveats as likelihood-based PF methods. A recent work [Ishizone et al., 2020] proposes blending VSMC and EnKF with an importance sampling-type lower bound estimate, which is effective if the state dimension is small. Other works that build on the VI framework include [Krishnan et al., 2017; Rangapuram et al., 2018; Fraccaro et al., 2017]. An important challenge is to obtain suitable parameterizations of the posterior, especially when the state dimension is high. For this reason, a restrictive Gaussian parameterization with a diagonal covariance matrix is often used in practice [Krishnan et al., 2017; Fraccaro et al., 2017].

More broadly, the development of data-driven ML frameworks for learning dynamical systems is a very active research area and we refer to [Brunton and Kutz, 2019; Gottwald and Reich, 2021; Harlim et al., 2021; Raissi et al., 2018] for recent references that illustrate a range of techniques that do not rely on the EM algorithm, autodiff of filtering methods, or VI.

Outline

This paper is organized as follows. §3.2 formalizes our framework and reviews a characterization of the likelihood in terms of normalizing constants arising in sequential filtering. §3.3 overviews EnKF algorithms for filtering and log-likelihood estimation. §3.4 contains our main methodological contributions. Numerical experiments on linear-Gaussian and Lorenz-96 models are described in §3.5. We close in §3.6.

Notation

We denote by $t \in \{0, 1, \ldots, T\}$ a discrete time index and by $n \in \{1, \ldots, N\}$ a particle index. Time indices will be denoted with subscripts and particles with superscripts, so that x_t^n represents a generic particle at time t. We denote $x_{t_0:t_1} := \{x_t\}_{t=t_0}^{t_1}$ and $x^{n_1:n_2} := \{x^n\}_{n=n_0}^{n_1}$. The collection $x_{t_0:t_1}^{n_0:n_1}$ is defined similarly. The Gaussian density with mean m and covariance C evaluated at x is denoted by $\mathcal{N}(x; m, C)$. The corresponding Gaussian distribution is denoted by $\mathcal{N}(m, C)$. For square matrices A and B, we write $A \succ B$ if $A - B$ is positive definite, and $A \succeq B$ if $A - B$ is positive semi-definite. For $A \succeq 0$, we denote by $A^{1/2}$ the unique matrix $B \succeq 0$ such that $B^2 = A$. We denote by $|v|$ the 2-norm of a vector v, and by $|A|$ the Frobenius norm of a matrix A.

3.2 Problem Formulation

Let $x_{0:T}$ be a time-homogeneous Markov chain of hidden *states* $x_t \in \mathbb{R}^{d_x}$ with transition kernel $p_\theta(x_t|x_{t-1})$ parameterized by $\theta \in \mathbb{R}^{d_\theta}$. Let $y_{1:T}$ be observations of the state. We seek to learn the parameter θ and recover the state process $x_{0:T}$ from the observations $y_{1:T}$. In §3.2.1, we formalize our problem setting, emphasizing our main goal of learning unknown dynamical systems for improved DA. §3.2.2 describes how the log-likelihood $\mathcal{L}(\theta) = \log p_\theta(y_{1:T})$ can be written in terms of normalizing constants arising from sequential filtering.

This idea will be used in §3.3 to obtain EnKF estimates for $\mathcal{L}(\theta)$ and $\nabla_\theta \mathcal{L}(\theta)$, which are then employed in §3.4 to learn θ by gradient ascent.

3.2.1 Setting and Motivation

We consider the following state-space model (SSM)

$$
\begin{aligned}
&\text{(transition)} \qquad && x_t = F_\alpha(x_{t-1}) + \xi_t, \qquad && \xi_t \sim \mathcal{N}(0, Q_\beta), \qquad && 1 \le t \le T, && (3.2.1)\\
&\text{(observation)} \qquad && y_t = H x_t + \eta_t, \qquad && \eta_t \sim \mathcal{N}(0, R), \qquad && 1 \le t \le T, && (3.2.2)\\
&\text{(initialization)} \qquad && x_0 \sim p_0(x_0). && && && (3.2.3)
\end{aligned}
$$

The initial distribution p_0 and the matrices $H \in R^{d_y \times d_x}$ and $R \succ 0$ are assumed to be known. Nonlinear observations can be dealt with by augmenting the state. We further assume independence of all random variables x_0, $\xi_{1:T}$, and $\eta_{1:T}$. Finally, the transition kernel $p_\theta(x_t|x_{t-1}) = \mathcal{N}(x_t; F_\alpha(x_{t-1}), Q_\beta)$, parameterized by $\theta := \{\alpha, \beta\}$, is defined in terms of a deterministic map F_α and Gaussian additive noise. This kernel approximates an unknown state transition of the form

$$
x_t = F^*(x_{t-1}) + \xi_t, \qquad \xi_t \sim \mathcal{N}(0, Q^*), \qquad 1 \le t \le T, \tag{3.2.4}
$$

where $Q^* = 0$ if the true evolution of the state is deterministic. The parameter β allows us to estimate the possibly unknown Q^*. We consider three categories of unknown state transition F^*, leading to three types of learning problems:

(a) *Parameterized dynamics*: $F^* = F_{\alpha^*}$ is parameterized, but the true parameter α^* is unknown and needs to be estimated.

(b) *Fully-unknown dynamics*: F^* is fully unknown and α represents the parameters of a NN surrogate model F_α^{NN} for F^*. The goal is to find an accurate surrogate model F_α^{NN}.

(c) *Model correction*: F^* is unknown, but an inaccurate model $F_{\text{approx}} \approx F^*$ is available. Here α represents the parameters of a NN G_α^{NN} used to correct the inaccurate model. The goal is to learn α so that $F_\alpha := F_{\text{approx}} + G_\alpha^{\text{NN}}$ approximates F^* accurately.

In some applications, the map F^* may represent the flow between observations of an autonomous differential equation driving the state, i.e.

$$\frac{\mathrm{d}x}{\mathrm{d}s} = f^*(x), \qquad F^* : x(s) \mapsto x(s + \Delta_s), \tag{3.2.5}$$

where f^* is an unknown vector field and Δ_s is the time between observations. Then, the map F_α in (3.2.1) (resp. F_α^{NN}, F_{approx}, G_α^{NN}) will be similarly defined as the Δ_s-flow of a differential equation with vector field f_α (resp. f_α^{NN}, f_{approx}, g_α^{NN}). Once $\theta = \{\alpha, \beta\}$ is learned, the state $x_{0:T}$ can be recovered with a filtering algorithm using the transition kernel $p_\theta(x_t|x_{t-1})$. We will illustrate the implementation and performance of AD-EnKF in these three categories of unknown dynamics in §3.5 using the Lorenz-96 model to define the vector field f^*. We remark that learning NN surrogate models for the dynamics may be useful even when the true state transition F^* is known, since F_α^{NN} may be cheaper to evaluate than F^*.

Our problem setting does not require to have access to a prior distribution on the parameter θ. If prior information is available, the AD-EnKF framework can seamlessly incorporate it replacing the log-likelihood with the log-posterior density in our subsequent developments. A Bayesian treatment can be appealing for unknown parameterized dynamics, where it is natural to have *a priori* information on the parameter. However, prior specification can be challenging for NN surrogate models.

3.2.2 Sequential Filtering and Data Log-likelihood

Suppose that $\theta = \{\alpha, \beta\}$ is known. We recall that, for $1 \le t \le T$, the *filtering distributions* $p_\theta(x_t|y_{1:t})$ of the SSM (3.2.1)-(3.2.2)-(3.2.3) can be obtained sequentially, alternating between *forecast* and *analysis* steps:

$$\text{(forecast)} \qquad p_\theta(x_t|y_{1:t-1}) = \int \mathcal{N}(x_t; F_\alpha(x_{t-1}), Q_\beta) p_\theta(x_{t-1}|y_{1:t-1}) \mathrm{d}x_{t-1}, \tag{3.2.6}$$

$$\text{(analysis)} \qquad p_\theta(x_t|y_{1:t}) = \frac{1}{Z_t(\theta)} \mathcal{N}(y_t; Hx_t, R) p_\theta(x_t|y_{1:t-1}), \tag{3.2.7}$$

with the convention $p_\theta(\cdot|y_{1:0}) := p_\theta(\cdot)$. Here $Z_t(\theta)$ is a normalizing constant which does not depend on x_t. It can be easily shown that

$$Z_t(\theta) = p_\theta(y_t|y_{1:t-1}) = \int \mathcal{N}(y_t; Hx_t, R) p_\theta(x_t|y_{1:t-1}) \mathrm{d}x_t, \tag{3.2.8}$$

and therefore the data log-likelihood admits the characterization

$$\mathcal{L}(\theta) := \log p_\theta(y_{1:T}) = \sum_{t=1}^{T} \log p_\theta(y_t|y_{1:t-1}) = \sum_{t=1}^{T} \log Z_t(\theta). \tag{3.2.9}$$

Analytical expressions of the filtering distributions $p_\theta(x_t|y_{1:t})$ and the data log-likelihood $\mathcal{L}(\theta)$ are only available for a small class of SSMs, which includes linear-Gaussian and discrete SSMs [Kalman, 1960; Papaspiliopoulos et al., 2014]. Outside these special cases, filtering algorithms need to be employed to approximate the filtering distributions, and these algorithms can be leveraged to estimate the log-likelihood.

3.3 Ensemble Kalman Filter Estimation of the Log-likelihood and its Gradient

In this section, we briefly review EnKFs and how they can be used to obtain an estimate $\mathcal{L}_{\text{EnKF}}(\theta)$ of the log-likelihood $\mathcal{L}(\theta)$. As will be detailed in §3.4, the map $\theta \mapsto \mathcal{L}_{\text{EnKF}}(\theta)$ can be readily auto-differentiated to compute $\nabla_\theta \mathcal{L}_{\text{EnKF}}(\theta)$, and this gradient can be used to learn the parameter θ. §3.3.1 gives background on EnKFs, §3.3.2 shows how EnKFs can be used to estimate $\mathcal{L}(\theta)$, and §3.3.3 contains novel convergence guarantees for the EnKF estimation of $\mathcal{L}(\theta)$ and $\nabla_\theta \mathcal{L}(\theta)$.

3.3.1 Ensemble Kalman Filters

Given $\theta = \{\alpha, \beta\}$, the EnKF algorithm [Evensen, 1994, 2009] sequentially approximates the filtering distributions $p_\theta(x_t|y_{1:t})$ using N equally-weighted particles $x_t^{1:N}$. At forecast steps, each particle x_t^n is propagated using the state transition equation Eq. (3.2.1), while at analysis steps a Kalman-type update is performed for each particle:

$$\text{(forecast step)} \qquad \widehat{x}_t^n = F_\alpha(x_{t-1}^n) + \xi_t^n, \qquad \xi_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, Q_\beta), \tag{3.3.1}$$

$$\text{(analysis step)} \qquad x_t^n = \widehat{x}_t^n + \widehat{K}_t(y_t + \gamma_t^n - H\widehat{x}_t^n), \qquad \gamma_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, R). \tag{3.3.2}$$

Note that the particles $x_{0:T}^{1:N}$ depend on θ, and (3.3.1)-(3.3.2) implicitly define a map $\theta \mapsto x_{0:T}^{1:N}$. The Kalman gain $\widehat{K}_t := \widehat{C}_t H^\top (H\widehat{C}_t H^\top + R)^{-1}$ is defined using the empirical covariance $\widehat{C}_t$ of the forecast ensemble $\widehat{x}_t^{1:N}$, namely

$$\widehat{C}_t = \frac{1}{N-1}\sum_{n=1}^N (\widehat{x}_t^n - \widehat{m}_t)(\widehat{x}_t^n - \widehat{m}_t)^\top, \quad \text{where} \quad \widehat{m}_t = \frac{1}{N}\sum_{n=1}^N \widehat{x}_t^n. \tag{3.3.3}$$

These empirical moments provide a Gaussian approximation to the *forecast distribution*

$$p_\theta(x_t|y_{1:t-1}) \approx \mathcal{N}(\widehat{m}_t, \widehat{C}_t). \tag{3.3.4}$$

Several implementations of EnKF are available, but for concreteness we only consider the "perturbed observation" EnKF defined in (3.3.1)-(3.3.2). In the analysis step (3.3.2), the observation y_t is perturbed to form $y_t + \gamma_t^n$. This perturbation ensures that in linear-Gaussian models the empirical mean and covariance of $x_t^{1:N}$ converges as $N \to \infty$ to the mean and covariance of the filtering distribution [Le Gland et al., 2009; Law et al., 2016b].

3.3.2 Estimation of the Log-Likelihood and its Gradient

Note from (3.2.9) that in order to approximate $\mathcal{L}(\theta) = \log p_\theta(y_{1:T})$, it suffices to approximate $p_\theta(y_t|y_{1:t-1})$ for $1 \le t \le T$. Now, using (3.2.8) and the EnKF approximation (3.3.4) to the forecast distribution, we obtain

$$p_\theta(y_t|y_{1:t-1}) \approx \int \mathcal{N}(y_t; Hx_t, R)\mathcal{N}(x_t; \widehat{m}_t, \widehat{C}_t)\mathrm{d}x_t = \mathcal{N}\big(y_t; H\widehat{m}_t, H\widehat{C}_t H^\top + R\big). \tag{3.3.5}$$

Therefore, we have the following estimate of the data log-likelihood:

$$\mathcal{L}_{\mathrm{EnKF}}(\theta) := \sum_{t=1}^{T} \log \mathcal{N}\big(y_t; H\widehat{m}_t, H\widehat{C}_t H^\top + R\big) \approx \mathcal{L}(\theta). \tag{3.3.6}$$

Notice that the forecast empirical moments $\{\widehat{m}_t, \widehat{C}_t\}_{t=1}^T$, and hence $\mathcal{L}_{\mathrm{EnKF}}(\theta)$, depend on θ in two distinct ways. First, each forecast particle $\widehat{x}_t^n$ in (3.3.1) depends on a particle x_t^n, which indirectly depends on θ. Second, each forecast particle depends on $\theta = \{\alpha, \beta\}$ directly through F_α and Q_β. The estimate $\mathcal{L}_{\mathrm{EnKF}}(\theta)$ can be computed online with EnKF, and is stochastic as it depends on the randomness used to propagate the particles, e.g., the choice of random seed. The whole procedure is summarized in Algorithm 3.3.1, which implicitly defines a stochastic map $\theta \mapsto \mathcal{L}_{\mathrm{EnKF}}(\theta)$. Before discussing the autodiff of this map and learning of the parameter θ in §3.4, we establish the large ensemble convergence of $\mathcal{L}_{\mathrm{EnKF}}(\theta)$ and $\nabla_\theta \mathcal{L}_{\mathrm{EnKF}}(\theta)$ towards $\mathcal{L}(\theta)$ and $\nabla_\theta \mathcal{L}(\theta)$ in a linear setting.

3.3.3 Large Sample Convergence: Linear Setting

In this section we consider a linear setting and provide large N convergence results for the log-likelihood estimate $\mathcal{L}_{\mathrm{EnKF}}(\theta)$ and its gradient $\nabla_\theta \mathcal{L}_{\mathrm{EnKF}}(\theta)$ towards $\mathcal{L}(\theta)$ and $\nabla_\theta \mathcal{L}(\theta)$ for any given θ, for a fixed data sequence $y_{1:T}$. The mappings $\mathcal{L}$ and $\mathcal{L}_{\mathrm{EnKF}}$ are defined in Eq. (3.2.9) and Eq. (3.3.6), respectively. For notation convenience, we drop θ in the function

Algorithm 3.3.1 Ensemble Kalman Filter and Log-likelihood Estimation

```
   Input: $\theta = \{\alpha, \beta\}, y_{1:T}, x_0^{1:N}$. (If $x_0^{1:N}$ is not specified, draw $x_0^n \overset{\text{i.i.d.}}{\sim} p_0(x_0)$.)
1: Initialize $\mathcal{L}_{\text{EnKF}}(\theta) = 0$.
2: for $t = 1, \dots, T$ do
3:     Set $\widehat{x}_t^n = F_\alpha(x_{t-1}^n) + \xi_t^n$, where $\xi_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, Q_\beta)$.        ▷ Forecast step
4:     Compute $\widehat{m}_t, \widehat{C}_t$ by Eq. (3.3.3) and set $\widehat{K}_t = \widehat{C}_t H^\top (H\widehat{C}_t H^\top + R)^{-1}$.
5:     Set $x_t^n = \widehat{x}_t^n + \widehat{K}_t(y_t + \gamma_t^n - H\widehat{x}_t^n)$, where $\gamma_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, R)$.        ▷ Analysis step
6:     Set $\mathcal{L}_{\text{EnKF}}(\theta) \leftarrow \mathcal{L}_{\text{EnKF}}(\theta) + \log \mathcal{N}\big(y_t; H\widehat{m}_t, H\widehat{C}_t H^\top + R\big)$.
7: end for
   Output: EnKF particles $x_{0:T}^{1:N}$. Log-likelihood estimate $\mathcal{L}_{\text{EnKF}}(\theta)$.
```

argument, since the main dependence will be on N in this section. Similar to [Le Gland et al., 2009; Kwiatkowski and Mandel, 2015], we study L^p convergence for any $p \geq 1$.

Theorem 3.3.1. *Assume that the state transition Eq.* (3.2.1) *is linear, i.e.,*

$$x_t = A_\alpha x_{t-1} + \xi_t, \qquad \xi_t \sim \mathcal{N}(0, Q_\beta), \qquad A_\alpha \in \mathbb{R}^{d_x \times d_x}, \tag{3.3.7}$$

and that the initial distribution p_0 is Gaussian. Then, for any $\theta = \{\alpha, \beta\}$ and for any $p \geq 1$, $\mathcal{L}_{EnKF}$ converges to $\mathcal{L}$ in L^p with rate $1/\sqrt{N}$, i.e.,

$$\big(\mathbb{E}\,|\mathcal{L}_{EnKF} - \mathcal{L}|^p\big)^{1/p} \leq cN^{-1/2}, \tag{3.3.8}$$

where c does not depend on N but may depend on θ, d_x and d_y.

The linearity of the flow $F_\alpha(\cdot)$ is equivalent to the linearity of the vector field $f_\alpha(\cdot)$. Although the convergence of the EnKF to the KF in linear settings has been studied in DA [Le Gland et al., 2009; Law et al., 2016b; Kwiatkowski and Mandel, 2015; Del Moral et al., 2018] and in filtering approaches to inverse problems [Schillings and Stuart, 2017; Chada et al., 2021], there are no existing convergence results for EnKF log-likelihood estimation. Two related works are [Katzfuss et al., 2020], which provides a heuristic argument for convergence in the case $T = 1$, and [Crisan et al., 2021], where a continuous-time version of

EnKF is considered.

Most of the theoretical analysis of EnKF is based on the *propagation of chaos* statement [McKean, 1967; Sznitman, 1991]: EnKF defines an interacting particle system, where the interaction is through the empirical mean $\widehat{m}_t$ and covariance matrix $\widehat{C}_t$ of the forecast ensemble $\widehat{x}_t^{1:N}$. As $N \to \infty$, one hopes that these empirical moments can be replaced by their deterministic limits, and that the particles will hence evolve independently. The large N limits of $\widehat{m}_t, \widehat{C}_t$ turn out to be the mean and covariance matrix of the KF forecast distribution. We will leave the construction of the propagation of chaos statement as well as the proof of Theorem 3.3.1 to §3.7.1.

Since this paper focuses on gradient based approaches to the learning of $\theta = \{\alpha, \beta\}$, it is thus interesting to compare the gradient $\nabla_\theta \mathcal{L}_{\mathrm{EnKF}}$ to the true gradient $\nabla_\theta \mathcal{L}$, as $N \to \infty$, if both of them exist. The intuition is that if $\nabla_\theta \mathcal{L}_{\mathrm{EnKF}}$ is an accurate estimate of $\nabla_\theta \mathcal{L}$, then one can perform gradient-based optimization over $\mathcal{L}_{\mathrm{EnKF}}$ as if one was directly optimizing over the true log-likelihood $\mathcal{L}$. For the gradient w.r.t. β to be well-defined, we write $S_\beta = Q_\beta^{1/2}$ in the following statement, so that β does not appear in the stochasticity of the algorithm. This is also known as the "reparameterization trick," which will be discussed later in §3.4.1.

Theorem 3.3.2. *Assume that the state transition Eq.* (3.2.1) *is linear, i.e.,*

$$x_t = A_\alpha x_{t-1} + S_\beta \xi_t, \qquad \xi_t \sim \mathcal{N}(0, I_{d_x}), \qquad A_\alpha \in \mathbb{R}^{d_x \times d_x}, \tag{3.3.9}$$

and that the initial distribution p_0 is Gaussian. Assume the parameterizations $\alpha \mapsto A_\alpha$ and $\beta \mapsto S_\beta$ are differentiable. Then, for any $\theta = \{\alpha, \beta\}$, both $\nabla_\theta \mathcal{L}_{EnKF}$ and $\nabla_\theta \mathcal{L}$ exist and, for any $p \geq 1$, $\nabla_\theta \mathcal{L}_{EnKF}$ converges to $\nabla_\theta \mathcal{L}$ in L^p with rate $1/\sqrt{N}$, i.e.,

$$\left(\mathbb{E}\left|\nabla_\theta \mathcal{L}_{EnKF} - \nabla_\theta \mathcal{L}\right|^p\right)^{1/p} \leq c N^{-1/2}, \tag{3.3.10}$$

where c does not depend on N but may depend on θ, d_x and d_y.

An important observation is that θ only enters the objective function $\mathcal{L}_{\text{EnKF}}$ through the empirical mean $\widehat{m}_t$ and covariance matrix $\widehat{C}_t$ of the forecast ensemble. As $N \to \infty$, one hopes that these empirical moments can be replaced by their deterministic limits, and gradients based on these empirical moments can be replaced by gradients based on their deterministic limits. The gradients taken in the limits turn out to be those of the true log-likelihood $\mathcal{L}$. Again, the proof relies on the propagation of chaos statement and is left to §3.7.2.

Theorem 3.3.2 should be compared with log-likelihood gradient estimation with PFs. The paper [Corenflos et al., 2021] shows that the gradient $\nabla_\theta \mathcal{L}_{\text{PF}}$ of PF log-likelihood estimate is biased, even in the linear setting, if one ignores the gradient from resampling steps, which is the method used in practice [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018].

3.4 Auto-differentiable Ensemble Kalman Filters

This section contains our main methodological contributions. We introduce our AD-EnKF framework in §3.4.1. We then describe in §3.4.2 how to handle long observation data, i.e., large T, using TBPTT. In §3.4.3, we highlight how various techniques introduced for EnKF in the DA community, e.g., localization and covariance inflation, can be incorporated into our framework. Finally, §3.4.4 discusses the computational and memory costs.

3.4.1 Main Algorithm

Algorithm 3.4.1 Auto-differentiable Ensemble Kalman Filter (AD-EnKF)

```
   Input: Observations $y_{1:T}$. Learning rate $\eta$.
1: Initialize SSM parameter $\theta^0$ and set $k = 0$.
2: while not converging do
3:     $x_{0:T}^{1:N}, \mathcal{L}_{\text{EnKF}}(\theta^k) = \textsc{EnsembleKalmanFilter}(\theta^k, y_{1:T})$.        ▷ Alg. 3.3.1
4:     Compute $\nabla_\theta \mathcal{L}_{\text{EnKF}}(\theta^k)$ by auto-differentiating the map $\theta^k \mapsto \mathcal{L}_{\text{EnKF}}(\theta^k)$.
5:     Set $\theta^{k+1} = \theta^k + \eta \nabla_\theta \mathcal{L}_{\text{EnKF}}(\theta^k)$ and $k \leftarrow k + 1$.
6: end while
   Output: Learned SSM parameter $\theta^k$ and EnKF particles $x_{0:T}^{1:N}$.
```

Our core method is shown in Alg. 3.4.1, and our PyTorch implementation is at `https://github.com/ymchen0/torchEnKF`. The gradient of the stochastic map $\theta^k \mapsto \mathcal{L}_{\text{EnKF}}(\theta^k)$ can be evaluated using autodiff libraries [Paszke et al., 2019; Bradbury et al., 2018; Abadi et al., 2016]. More specifically, reverse-mode autodiff can be performed for common matrix operations like matrix multiplication, inverse, and determinant [Giles, 2008a]. We use the "reparameterization trick" [Kingma and Welling, 2014; Rezende et al., 2014] to auto-differentiate through the stochasticity in the EnKF algorithm. Specifically, in Alg. 3.3.1 line 3, we draw ξ_t^n from a distribution $\mathcal{N}(0, Q_\beta)$ that involves a parameter β with respect to which we would like to compute the gradient. For this operation to be compatible with the autodiff, we reparameterize

$$\widehat{x}_t^n = F_\alpha(x_t^n) + \xi_t^n \quad \xi_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, Q_\beta) \quad \Longleftrightarrow \quad \widehat{x}_t^n = F_\alpha(x_t^n) + Q_\beta^{1/2}\xi_t^n \quad \xi_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_{d_x}), \tag{3.4.1}$$

so that the gradient with respect to β admits an unbiased estimate. In contrast to the EnKF, the resampling step of PFs cannot be readily auto-differentiated [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018]. The algorithm can be stopped when certain convergence criteria have been met, e.g., when the relative change in the 10-step moving average of the EnKF log-likelihood $\mathcal{L}_{\text{EnKF}}(\theta^k)$ does not exceed a pre-specified threshold of 10^{-2}. In our numerical experiments in §3.5, we run the algorithm for at least 50 additional iterations past convergence to demonstrate its long-time performance and stability.

3.4.2 Truncated Gradients for Long Sequences

If the sequence length T is large, although $\mathcal{L}_{\text{EnKF}}(\theta)$ and its gradient $\nabla_\theta \mathcal{L}_{\text{EnKF}}(\theta)$ can be evaluated using the aforementioned techniques, the practical value of Alg. 3.4.1 is limited for two reasons. First, computing these quantities requires a full filtering pass of the data, which may be computationally costly. Moreover, for the gradient ascent methods to achieve a good convergence rate, multiple evaluations of gradients are often needed, requiring an equally large number of filtering passes. The second reason is that, like recurrent networks, Alg. 3.4.1 may suffer from exploding or vanishing gradients [Pascanu et al., 2013] as the derivatives are multiplied together using chain rules in the backpropagation.

Our proposed technique can address both of these issues by borrowing the ideas of TBPTT from the recurrent neural network literature [Williams and Zipser, 1995; Sutskever et al., 2014] and the recursive maximum likelihood method from the hidden Markov models literature [Le Gland and Mevel, 1997]. The idea is to divide the sequence into subsequences of length L. Instead of computing the log-likelihood of the whole sequence and then backpropagating, one computes the log-likelihood of each subsequence and backpropagates within that subsequence. The subsequences are processed sequentially, and the EnKF output of the previous subsequence (i.e., the location of particles) are used as the input to the next subsequence. In this way, one performs $\lceil T/L \rceil$ gradient updates in a *single* filtering pass, and since the gradients are backpropagated across a time span of length at most L, gradient explosion/vanishing is more unlikely to happen. This approach is detailed in Alg. 3.4.2.

Algorithm 3.4.2 AD-EnKF with Truncated Backprop (AD-EnKF-T)

```
   Input: Observations $y_{1:T}$. Learning rate $\eta$. Subsequence length $L$.
1: Initialize SSM parameter $\theta^0$ and set $k = 0$.
2: while not converging do
3:     Set $x_0^n \overset{\text{i.i.d.}}{\sim} p_0(x_0)$.
4:     for $j = 0, \dots, T/L - 1$ do
5:         Set $t_0 = jL$, $t_1 = \min\{(j+1)L, T\}$.
6:         $x_{t_0:t_1}^{1:N}, \mathcal{L}_{\text{EnKF}}(\theta^k) = \text{EnsembleKalmanFilter}(\theta^k, y_{(t_0+1):t_1}, x_{t_0}^{1:N})$. ▷ Alg. 3.3.1
7:         Set $\theta^{k+1} = \theta^k + \eta \nabla_\theta \mathcal{L}_{\text{EnKF}}(\theta^k)$ and $k \leftarrow k + 1$.
8:     end for
9: end while
   Output: Learned SSM parameter $\theta^k$ and EnKF particles $x_{0:T}^{1:N}$.
```

3.4.3 Localization for High State Dimensions

In practice, the state often represents a physical quantity that is discretized in spatial coordinates (e.g., numerical solution to a time-evolving PDE), which leads to a high state dimension d_x. In order to reduce the computational and memory complexity, EnKF is often run with $N < d_x$. A small ensemble size N causes rank deficiency of the forecast sample covariance $\widehat{C}_t$, which may cause spurious correlations between spatial coordinates that are far apart. In other words, for (i, j) such that $|i - j|$ is large, the (i, j)-th coordinate of $\widehat{C}_t$ may not be close to 0, although one would expect it to be small since it represents the correlation between spatial locations that are far apart. This problem can be addressed using localization techniques, and we shall focus on *covariance tapering* [Houtekamer and Mitchell, 1998]. The idea is to "taper" the forecast sample covariance matrix $\widehat{C}_t$ so that the nonzero spurious correlations are zeroed out. This method is implemented defining a $d_x \times d_x$ matrix ρ with 1's on the diagonal and entries smoothly decaying to 0 off the diagonal, and replacing the forecast sample covariance matrix $\widehat{C}_t$ in Alg. 3.3.1 by $\rho \circ \widehat{C}_t$, where $\circ$ denotes the element-wise matrix product. Common choices of ρ were introduced in [Gaspari and Cohn, 1999]. Covariance tapering can be easily adopted within our AD-EnKF framework. We find that covariance tapering not only stabilizes the filtering procedure, which had been noted before, e.g., [Houtekamer and Mitchell, 2001; Hamill et al., 2001], but it also helps to obtain low-variance estimates of the log-likelihood and its gradient — see the discussion in §3.5.1. Localization techniques relying on local serial updating of the state [Houtekamer and Mitchell, 2001; Ott et al., 2004; Sakov and Bertino, 2011] could also be considered.

Another useful tool for EnKF with $N < d_x$ is *covariance inflation* [Anderson and Anderson, 1999], which prevents the ensemble from collapsing towards its mean after the analysis update [Furrer and Bengtsson, 2007]. In practice, this can be performed by replacing the forecast sample covariance matrix $\widehat{C}_t$ in Alg. 3.3.1 by $(1 + \zeta)\widehat{C}_t$, where $\zeta > 0$ is a small constant that needs to be tuned. Although not considered in our experiments, covariance inflation can also be easily adopted within our AD-EnKF framework.

3.4.4 Computation and Memory Costs

Autodifferentiation of the map $\theta^k \mapsto \mathcal{L}_{\text{EnKF}}(\theta^k)$ in Alg. 3.4.1 does not introduce an extra order of computational cost compared to the evaluation of this map alone. Thus, the computational cost of AD-EnKF is at the same order as that of a standard EnKF. The computation cost of EnKF can be found in, e.g., [Roth et al., 2017]. Moreover, AD-EnKF can be parallelized and speeded up with a GPU.

Like a standard EnKF, when no covariance tapering is applied, AD-EnKF has $O(Nd_x)$ memory cost since it does not explicitly compute the sample covariance matrix $\widehat{C}_t$[1]. With covariance tapering, the memory cost is at most $O(\max\{N, r\}d_x)$, where r is the tapering radius, if the tapering matrix ρ is sparse with $O(rd_x)$ nonzero entries. This sparsity condition is satisfied when using common tapering matrices [Gaspari and Cohn, 1999]. In terms of the time dimension, the memory cost of AD-EnKF can be reduced from $O(T)$ to $O(L)$ with the TBPTT in §3.4.2. Unlike previous work on EM-based approaches [Brajard et al., 2020; Bocquet et al., 2020; Pulido et al., 2018], where the locations of all particles $x_t^{1:N}$ across the whole time span of T need to be stored, AD-EnKF-T only requires to store the particles within a time span of L to perform a gradient step.

If the transition map F_α is defined by the flow map of an ODE with vector field f_α, we can use adjoint methods to differentiate efficiently through F_α in the forecast step Eq. (3.3.1). Use of the adjoint method is facilitated by NeuralODE autodiff libraries [Chen et al., 2018] that have become an important tool to learn continuous-time dynamical systems [Ayed et al., 2019; Rubanova et al., 2019; De Brouwer et al., 2019]. Instead of discretizing F_α with a numerical solver applied to f_α and differentiating through solver's steps as in [Brajard

1. Note that $\widehat{C}_t H^\top = \frac{1}{N-1}\sum_{n=1}^N (\widehat{x}_t^n - \widehat{m}_t)(H\widehat{x}_t^n - H\widehat{m}_t)^\top$ and $H\widehat{C}_t H^\top = \frac{1}{N-1}\sum_{n=1}^N (H\widehat{x}_t^n - H\widehat{m}_t)(H\widehat{x}_t^n - H\widehat{m}_t)^\top$, which require $O(d_x \max\{N, d_y\})$ and $O(d_y \max\{N, d_y\})$ memory respectively. Both of them are less than $O(d_x^2)$ if $d_y \ll d_x$ and $N \ll d_x$.

et al., 2020; Bocquet et al., 2020], we directly differentiate through F_α by solving an adjoint differential equation, which does not require us to store all intermediate steps from the numerical solver, reducing the memory cost. More details can be found in [Chen et al., 2018], and the PyTorch package provided by the authors can be incorporated within our AD-EnKF framework with minimal effort.

3.5 Numerical Experiments

3.5.1 Linear-Gaussian Model

In this section, we focus on parameter estimation in a linear-Gaussian model with a banded structure on model dynamic and model error covariance matrix. This experiment falls into the category of "parameterized dynamics" in §3.2.1. We first illustrate the convergence results of the log-likelihood estimate $\mathcal{L}_{\text{EnKF}}$ and gradient estimate $\nabla_\theta \mathcal{L}_{\text{EnKF}}$ presented in §3.3.3, since the true values $\mathcal{L}$ and $\nabla_\theta \mathcal{L}$ are available in closed form. We also show that the localization techniques described in §3.4.3 lead to a more accurate estimate when the ensemble size is small. Finally, we show that having a more accurate estimate, especially for the gradient, improves the parameter estimation.

We compare the EnKF to PF methods. Similar to the EnKF, the PF also provides an estimate of the log-likelihood and its gradient. Different from [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018], we adopt the PF with optimal proposal [Doucet and Johansen, 2009] as it is implementable for the family of SSMs considered in this paper [Doucet et al., 2000; Sanz-Alonso et al., 2018], and we find it to be more stable than separately training a variational proposal. To compute the log-likelihood gradient for the PF, we follow the same strategy as in [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018] and do not differentiate through the resampling step. The full algorithm, which we abbreviate as AD-PF, is presented in §3.7.4.

We consider the following SSM, similar to [Xu and Wikle, 2007; Stroud et al., 2018]

$$x_t = A_\alpha x_{t-1} + \xi_t, \qquad \xi_t \sim \mathcal{N}(0, Q_\beta), \qquad 1 \le t \le T, \tag{3.5.1a}$$

$$y_t = H x_t + \eta_t, \qquad \eta_t \sim \mathcal{N}(0, 0.5 I_{d_y}), \qquad 1 \le t \le T, \tag{3.5.1b}$$

$$x_0 \sim \mathcal{N}(0, 4 I_{d_x}), \tag{3.5.1c}$$

where

$$A_\alpha = \begin{bmatrix} \alpha_1 & \alpha_2 & & 0 \\ \alpha_3 & \alpha_1 & \ddots & \\ & \ddots & \ddots & \alpha_2 \\ 0 & & \alpha_3 & \alpha_1 \end{bmatrix}, \qquad [Q_\beta]_{i,j} = \beta_1 \exp(-\beta_2 |i-j|). \tag{3.5.1d}$$

Here $[Q_\beta]_{i,j}$ denotes the (i, j)-th entry of Q_β. Intuitively, β_1 controls the scale of error, while β_2 controls how error is correlated across spatial coordinates. We set $\alpha = (\alpha_1, \alpha_2, \alpha_3)$, $\beta = (\beta_1, \beta_2)$, and $\theta = \{\alpha, \beta\}$.

Estimation Accuracy of $\mathcal{L}_{\text{EnKF}}$ and $\nabla_\theta \mathcal{L}_{\text{EnKF}}$

As detailed above, a key idea proposed in this paper is to estimate $\mathcal{L}(\theta)$, $\nabla_\alpha \mathcal{L}(\theta)$ and $\nabla_\beta \mathcal{L}(\theta)$ with quantities $\mathcal{L}_{\text{EnKF}}(\theta)$, $\nabla_\alpha \mathcal{L}_{\text{EnKF}}(\theta)$ and $\nabla_\beta \mathcal{L}_{\text{EnKF}}(\theta)$ obtained by running an EnKF and differentiating through its computations using autodiff. Since these estimates will be used by AD-EnKF to perform gradient ascent, it is critical to assess their accuracy. We do so in this section for a range of values of θ.

We first simulate observation data $y_{1:T}$ from the true model with $d_x = d_y \in \{20, 40, 80\}$, $T = 10$, $H = I_{d_x}$, $\alpha^* = (0.3, 0.6, 0.1)$ and $\beta^* = (0.5, 1)$. Given data $y_{1:T}$, the true data log-likelihood $\mathcal{L}(\theta) = p_\theta(y_{1:T})$ and gradient $\nabla_\theta \mathcal{L}(\theta)$, which can be decomposed into $\nabla_\alpha \mathcal{L}(\theta)$, $\nabla_\beta \mathcal{L}(\theta)$, can be computed analytically. We perform $P = 50$ EnKF runs, and report a Monte Carlo estimate of the relative L^2 errors of the log-likelihood and gradient estimates (see

§3.7.6 for their definition) as the ensemble size N increases. Fig. 3.2 shows the results when θ is evaluated at the true parameters $\{\alpha^*, \beta^*\}$. Intuitively, this θ is close to optimal since it is the one that generates the data. We also show in Fig. 3.10 in §3.7.3 the results when θ is evaluated at a parameter that is not close to optimal: $\alpha = (0.5, 0.5, 0.5), \beta = (1, 0.1)$. Both figures illustrate that the relative L^2 estimation errors of the log-likelihood and its gradient computed using EnKF converge to zero at a rate of approximately $N^{-1/2}$. Moreover, the state dimension d_x has small empirical effect on the convergence rate. On the other hand, those computed using PF have a slower convergence rate or barely converge, especially for the gradient (see the third plot in Fig. 3.10). We recall that the resampling parts are discarded from the autodiff of PFs, which introduces a bias. Moreover, the empirical convergence rate is slightly slower in higher state dimensions. Comparing the estimation error of EnKF and PF under the same d_x choice, we find that when the number of particles is large (> 500), EnKF gives a more accurate estimate than PF. However, when the number of particles is small, EnKF is less accurate, but we will show in the next section how the EnKF results can be significantly improved using localization techniques. Unreported experimental results suggest that the relative L^2 error in the EnKF estimation of the log-likelihood and its gradient increase linearly with T for a fixed ensemble size N.

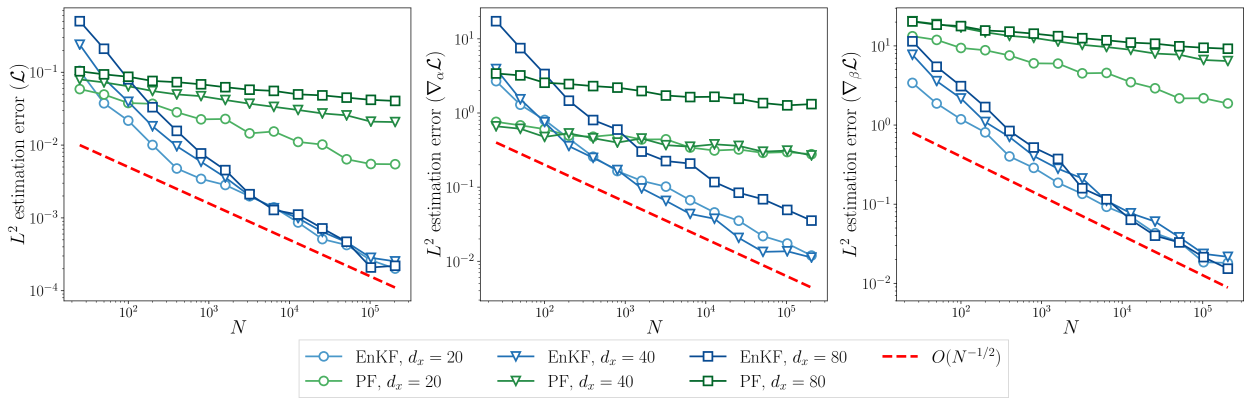


Figure 3.2: Relative L^2 estimation errors of the log-likelihood (left) and its gradient w.r.t. α (middle) and β (right), computed using EnKF and PF, as a function of N, for the linear-Gaussian model (3.5.1). State dimension $d_x \in \{20, 40, 80\}$. θ is evaluated at the true parameters $\{\alpha^*, \beta^*\}$. (§3.5.1).

Effect of Localization

In practice, for computational and memory concerns, the number of particles used for EnKF is typically small (< 100), and hence it is necessary to get an accurate estimate of log-likelihood and its gradients using a small number of particles. We use the covariance tapering techniques discussed in §3.4.3, where $\widehat{C}_t$ is replaced by $\rho \circ \widehat{C}_t$ in Alg. 3.3.1, and ρ is defined using the fifth order piecewise polynomial correlation function of Gasperi and Cohn [Gaspari and Cohn, 1999]. The detailed construction of ρ is left to §3.7.6, with a hyperparameter r that controls the tapering radius.

Fig. 3.3 shows the estimation results when the state dimension is set to be $d_x = 80$ and θ is evaluated at (α^*, β^*), while different tapering radii r are applied. The plots of EnKF with no tapering and the plots of PF are the same as in Fig. 3.2. We find that covariance tapering can reduce the estimation error of the log-likelihood and its gradient when the number of particles is small. Moreover, having a smaller tapering radius leads to a better estimation when the number of particles is small. As the number of particles grows larger, covariance tapering may worsen the estimation of both log-likelihood and its gradient. This is because the sampling error and spurious correlation that occurs in the sample covariance matrix in EnKF will be overcome by a large number of particles, and hence covariance tapering will only act as a modification to the objective function $\mathcal{L}_{\text{EnKF}}$, leading to inconsistent estimates. However, there is no reason for using localization when one can afford a large number of particles. When computational constraints require fewer particles than state dimension, we find that covariance tapering is not only beneficial to the parameter estimation problems but is also beneficial to learning of the dynamics in high dimensions, as we will show in later sections. Results when θ is evaluated at parameters that are not optimal ($\alpha = (0.5, 0.5, 0.5)$, $\beta = (1, 0.1)$) are shown in Fig. 3.11 in §3.7.3, where the beneficial effect of tapering is evident.

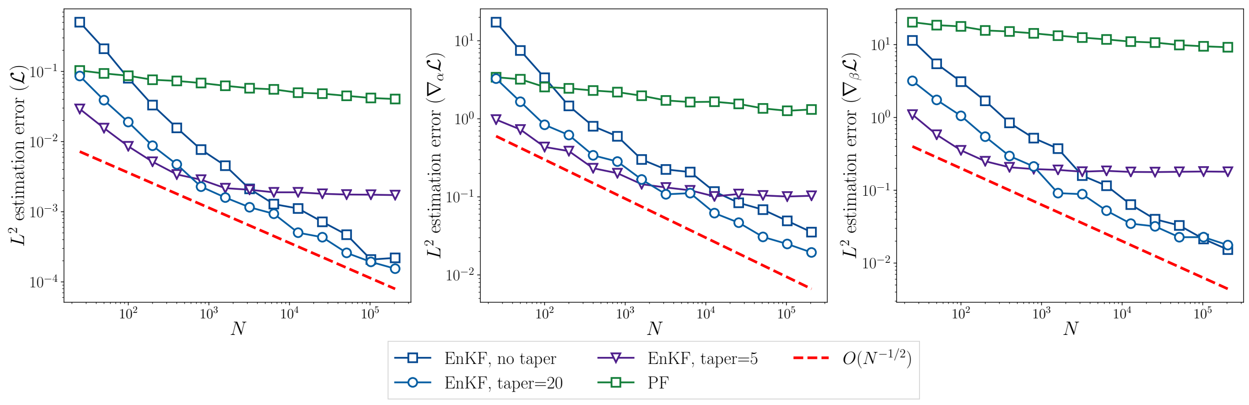


Figure 3.3: Relative L^2 estimation errors of log-likelihood (left) and its gradient w.r.t. α (middle) and β (right), computed using EnKF and PF, with different covariance tapering radius applied to EnKF for the linear-Gaussian model (3.5.1). State dimension $d_x = 80$. θ is evaluated at the true parameters $\{\alpha^*, \beta^*\}$. (§3.5.1).

Parameter Learning

Here we illustrate how the estimation accuracy of the log-likelihood and its gradient, especially the latter, affect the parameter learning with AD-EnKF. Since our framework relies on gradient-based learning of parameters, intuitively, the less biased the gradient estimate is, the closer our learned parameter will be to the true MLE solution.

We first consider the setting where the state dimension is set to be $d_x = 80$. We run AD-EnKF for 1000 iterations with gradient ascent under the following choices of ensemble size and tapering radius: (1) $N = 1000$ with no tapering; (2) $N = 50$ with no tapering; and (3) $N = 50$ with tapering radius 5. We also run AD-PF with $N = 1000$ particles. Throughout, one “training iteration” corresponds to processing the whole data sequence once. Additional implementation details are available in the appendices. Fig. 3.4 and 3.5 show a single run of parameter learning under each setting, where we include for reference the MLE obtained by running gradient ascent until convergence with the true gradient $\nabla_\theta \mathcal{L}$ (denoted with the red dashed line). The objective function, i.e., the likelihood estimates $\mathcal{L}_{\text{EnKF}}$ and $\mathcal{L}_{\text{PF}}$ are also plotted as a function of training iterations. Results with other choices of state dimension d_x are summarized in Table 3.5.1, where we take the values of α at the final iteration and compute their distance to the true MLE solution. The procedure is repeated 10 times, and the mean and standard deviations are reported. The results all show a similar trend: AD-EnKF with $N = 1000$ particles performs the best (small errors and small fluctuations) for all settings, while AD-EnKF with $N = 50$ particles and covariance tapering performs second best. AD-EnKF with $N = 50$ without covariance tapering comes at the third place, and AD-PF method performs the worst, indicating the superiority of AD-EnKF method to the AD-PF method for high-dimensional linear-Gaussian models of the form (3.5.1) and the utility of localization techniques. Importantly, the findings here are consistent with the plots in Fig. 3.3. This behavior is in agreement with the intuition that the estimation accuracy of the log-likelihood gradient determines the parameter learning performance.

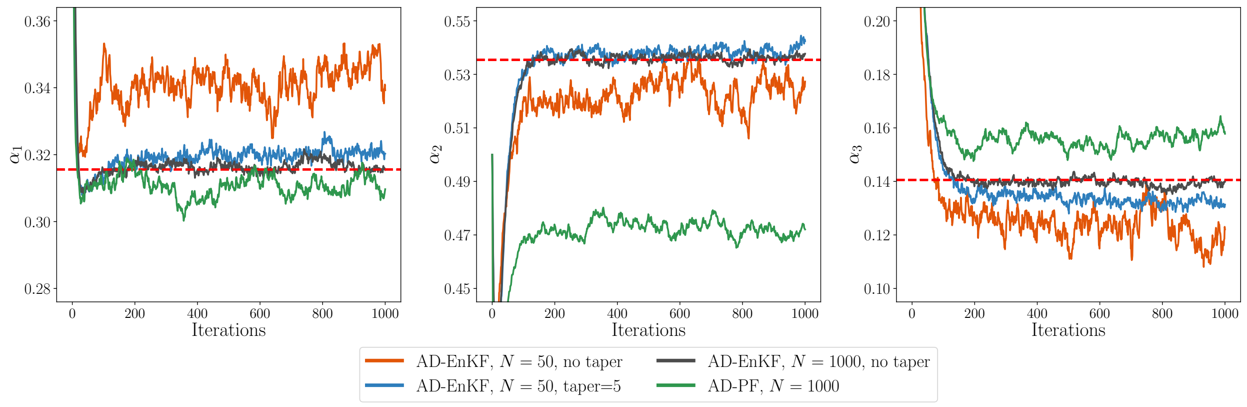


Figure 3.4: Learned parameter α as a function of training iterations for the linear-Gaussian model (3.5.1). State dimension $d_x = 80$. Red dashed lines are the MLE solutions to the true data log-likelihood $\mathcal{L}$. Our proposed AD-EnKF method with covariance tapering achieves a lower estimation error with $N = 50$ particles than AD-PF with $N = 1000$.

(§3.5.1).

3.5.2 Lorenz-96

In this section, we illustrate our AD-EnKF framework in the three types of learning problems mentioned in §3.2.1: parameterized dynamics, fully-unknown dynamics, and model correction. We will compare our method to AD-PF, as in §3.5.1. We will also compare our method to the EM-EnKF method implemented in [Bocquet et al., 2020; Brajard et al., 2020], which

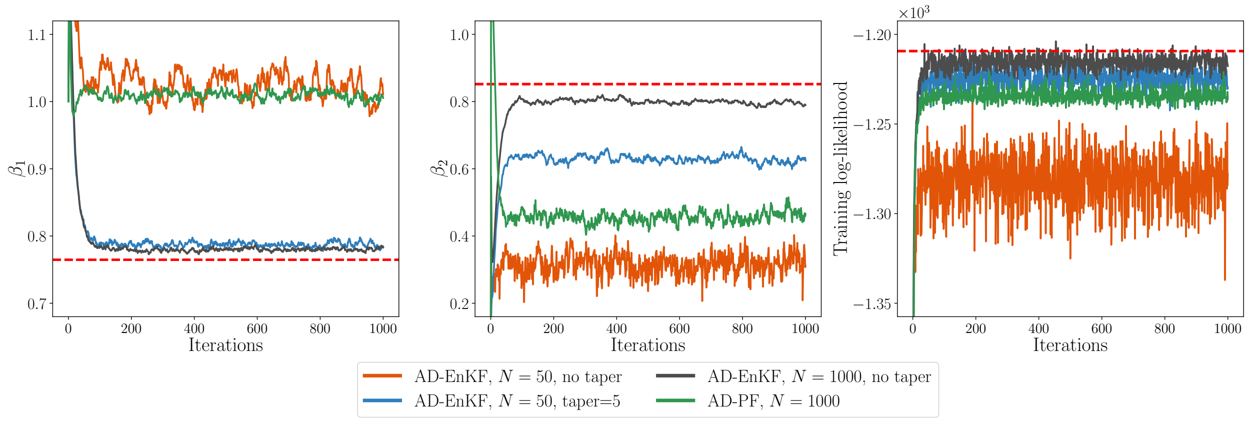


Figure 3.5: Learned parameter β, and training objective $\mathcal{L}_{\text{EnKF}}$, $\mathcal{L}_{\text{PF}}$ as a function of training iterations for the linear-Gaussian model (3.5.1). Red dashed lines are the MLE solutions to the true data log-likelihood $\mathcal{L}$ (left and middle), and the maximum value attained by $\mathcal{L}$ (right). Our proposed AD-EnKF method with covariance tapering achieves a lower estimation error with $N = 50$ particles than AD-PF with $N = 1000$.

(§3.5.1).

	$d_x = 20$ $N = 50$	$d_x = 20$ $N = 1000$	$d_x = 40$ $N = 50$	$d_x = 40$ $N = 1000$	$d_x = 80$ $N = 50$	$d_x = 80$ $N = 1000$
AD-EnKF (no taper)	1.65 ±0.30	0.07 ±0.06	4.12±0.73	0.17±0.09	4.14±0.67	0.20±0.14
AD-EnKF(taper=5)	0.53±0.18	–	0.35±0.27	–	1.05±0.38	–
AD-PF	7.75±0.37	3.51±0.35	8.58±0.25	5.59±0.31	9.28±0.49	6.77±0.24

Table 3.5.1: Euclidean distance ($\times 10^{-2}$) from the learned parameter α at the final iteration to the true MLE solution, under varying dimensional settings for the linear-Gaussian model (3.5.1). The parameter values recovered by our proposed AD-EnKF method with covariance tapering and $N = 50$ are closer to the MLE solution than the ones recovered by AD-PF with $N = 1000$.

(§3.5.1).

we abbreviate as EM, and is detailed in §3.7.5. We emphasize that the gradients computed in the EM are different from the ones computed in AD-EnKF, and in particular do not auto-differentiate through the EnKF.

The reference Lorenz-96 model [Lorenz, 1996] is defined by (3.2.5) with vector field

$$f^{*(i)}(x) = -x^{(i-1)}(x^{(i-2)} - x^{(i+1)}) - x^{(i)} + 8, \qquad 0 \le i \le d_x - 1, \tag{3.5.2}$$

where $x^{(i)}$ and $f^{*(i)}$ are the i-th coordinate of x and component of f^*. By convention $x^{(-1)} := x^{(d_x-1)}, x^{(-2)} := x^{(d_x-2)}$ and $x^{(d_x)} := x^{(0)}$. We assume there is no noise in the reference state transition model, i.e., $Q^* = 0$. The goal is to recover the reference state transition model with $p_\theta(x_t|x_{t-1}) = \mathcal{N}\big(x_t; F_\alpha(x_{t-1}), Q_\beta\big)$ from the data $y_{1:T}$, where F_α is the flow map of a vector field f_α, and then recover the states $x_{1:T}$. The parameterized error covariance Q_β in the transition model is assumed to be diagonal, i.e., $Q_\beta = \operatorname{diag}(\beta)$ with $\beta \in \mathbb{R}^{d_x}$. The parameterized vector field f_α is defined differently for the three types of learning problems, as we lay out below. We quantify performance using the forecast error (RMSE-f), the analysis/filter error (RMSE-a), and the test log-likelihood. These metrics are defined in §3.7.6.

Parameterized Dynamics

We consider the same setting as in [Bocquet et al., 2019], where

$$\begin{aligned} f_\alpha^{(i)}(x) = \big[&1, x^{(i-2)}, x^{(i-1)}, x^{(i)}, x^{(i+1)}, x^{(i+2)}, \\ &\big(x^{(i-2)}\big)^2, \big(x^{(i-1)}\big)^2, \big(x^{(i)}\big)^2, \big(x^{(i+1)}\big)^2, \big(x^{(i+2)}\big)^2, \\ &x^{(i-2)}x^{(i-1)}, x^{(i-1)}x^{(i)}, x^{(i)}x^{(i+1)}, x^{(i+1)}x^{(i+2)}, \\ &x^{(i-2)}x^{(i)}, x^{(i-1)}x^{(i+1)}, x^{(i)}x^{(i+2)}\big]^\top \alpha, \qquad 0 \le i \le d_x - 1, \end{aligned} \tag{3.5.3}$$

and $\alpha \in \mathbb{R}^{18}$ is interpreted as the coefficients of some "basis polynomials" representing the governing equation of the underlying system. The parameterized governing equation of the i-th coordinate depends on its $N_1 = 5$ neighboring coordinates, and the second order polynomials only involve interactions between coordinates that are at most $N_2 = 2$ indices apart. The reference ODE Eq. (3.5.2) satisfies $f^* = f_{\alpha^*}$, where $\alpha^* \in \mathbb{R}^{18}$ has nonzero entries

$$\alpha_0^* = 8, \quad \alpha_3^* = -1, \quad \alpha_{11}^* = -1, \quad \alpha_{16}^* = 1, \tag{3.5.4}$$

and zero entries otherwise. Here the dimension of $\theta = \{\alpha, \beta\}$ is $d_\theta = 18 + d_x$.

We first consider the specific case with $d_x = d_y = 40$, $H = I_{40}$. We set $R = I_{40}$ and $x_0 \sim \mathcal{N}(0, 50 I_{40})$. We generate four sequences of training data with the reference model for $T = 300$ with time between consecutive observations $\Delta_s = 0.05$. Both flow maps F^* and F_α are integrated using a fourth-order Runge-Kutta (RK4) method with step size $\Delta_s^{\text{int}} = 0.01$, with adjoint methods implemented for backpropagation through the ODE solver [Chen et al., 2018].

We use AD-EnKF-T (Alg. 3.4.2) with $L = 20$ and covariance tapering Eq. (3.7.45) with radius $r = 5$. We compare with AD-PF-T (see §3.7.4) with $L = 20$ and EM (see §3.7.5). L is chosen from the set $\{1, 5, 10, 20, 50, 100\}$ with the lowest forecast RMSE on the test set at the final training iteration. The implementation details, including the choice of learning rates and other hyperparameters, are discussed in §3.7.6.

Comparison of the three algorithms is shown in Fig. 3.6. Our AD-EnKF-T recovers α^* better than the other two approaches. The EM approach converges faster, but has a larger error. Moreover, EM tends to converge to a higher level of learned model error σ_β (defined in Eq. (3.7.46)), while our AD-EnKF-T shows a consistent drop of learned error level. Note that Q_β in the learned transition kernel acts like covariance inflation, which is discussed in §3.4.3, but is "learned" to be adaptive to the training data rather than manually tuned; therefore, having a nonzero error level σ_β may still be helpful. The plot of the log-likelihood estimate during training indicates that AD-EnKF-T searches for parameters with a higher log-likelihood than the EM approach, which is not surprising as AD-EnKF-T directly optimizes $\mathcal{L}_{\text{EnKF}}$, while EM does so by alternatively optimizing a surrogate objective. Also, the large discrepancy between the optimized $\mathcal{L}_{\text{EnKF}}$ and $\mathcal{L}_{\text{PF}}$ objective may be due to $\mathcal{L}_{\text{PF}}$ being a worse estimate for the true log-likelihood $\mathcal{L}$ than that of $\mathcal{L}_{\text{EnKF}}$. Note that PFs may not be suitable for high-dimensional systems like the Lorenz-96 model. Even with knowledge of the true reference model and a large number of particles, the PF is not able to capture the filtering distribution well due to the high dimensionality — see, e.g., Figure 5 of [Bocquet et al., 2010]. The plots of forecast error, filter error and test log-likelihood are presented in Fig. 3.7.

We also consider varying the state dimension d_x and observation model H. (The parameterization in Eq. (3.5.3) is valid for any choice of d_x.) We measure the Euclidean distance between the value of learned α at the final training iteration (at convergence) to α^*. The training procedure is repeated 5 times and the results are shown in Table 3.5.2. We vary $d_x \in \{10, 20, 40, 80\}$ and consider two settings for H: fully observed at all coordinates, i.e., $H = I_{d_x}$, and partially observed at every two out of three coordinates [Sanz-Alonso and Stuart, 2015], i.e., $H = [e_1, e_2, e_4, e_5, e_7, \cdots]^\top$, where $\{e_i\}_{i=1}^{d_x}$ is the standard basis for $\mathbb{R}^{d_x}$. The number of particles used for all algorithms is fixed at $N = 50$, and covariance tapering Eq. (3.7.45) with radius $r = 5$ is applied to the EnKF. For both AD-EnKF-T and AD-PF-T, $L = 20$. We find that AD-EnKF-T is able to consistently recover α^* regardless of the choice of d_x and H, and is able to perform well in the important case where $N < d_x$, with an accuracy that is orders of magnitude better than the other two approaches. The EM approach is able to recover α^* consistently in fully observed settings, but with a lower accuracy. In partially-observed settings, EM does not converge to the same value in repeated runs, possibly due to the existence of multiple local maxima. AD-PF-T is able to converge consistently in fully observed settings but with the lowest accuracy, and runs into filter divergence issues

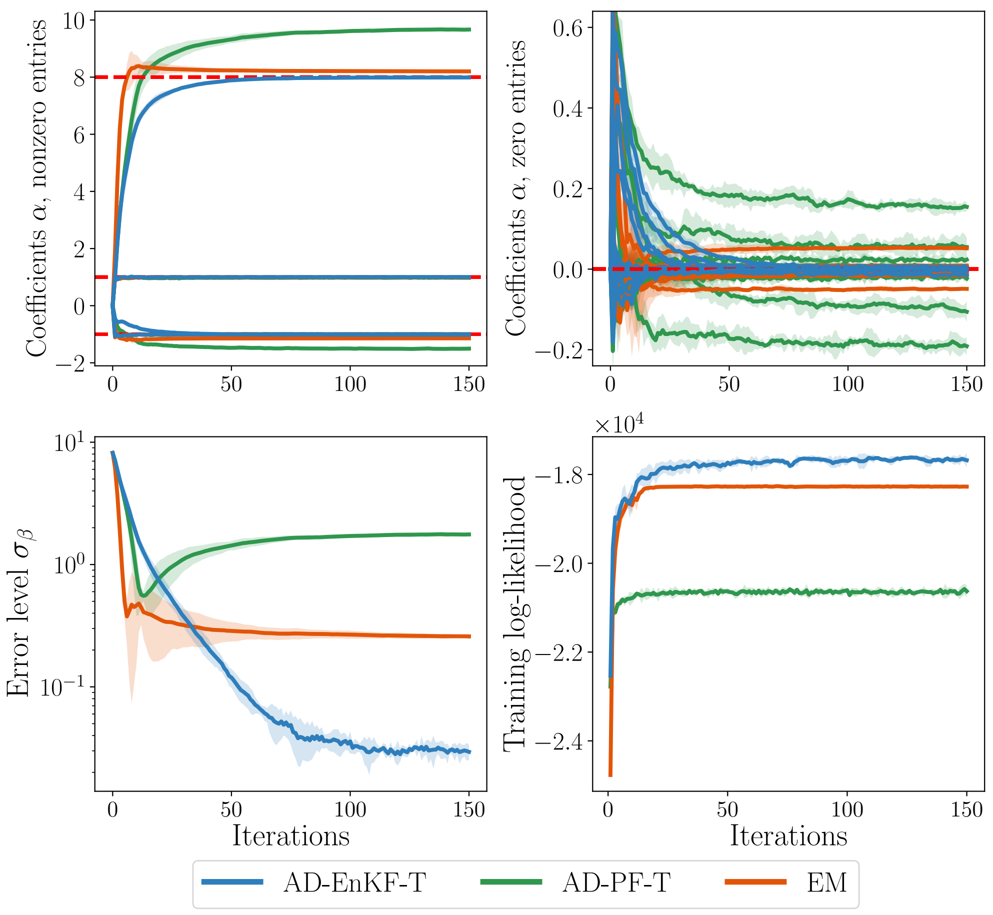


Figure 3.6: Learning parameterized dynamics of Lorenz-96 (3.5.3), with $d_x = 40$ and $H = I_{40}$. Learned value of the 18 coefficients of α (upper left for nonzero entries and upper right for zero entries, where the truth α^* is plotted in red dashed lines), averaged diagnosed error level σ_β Eq. (3.7.46) (lower left) and log-likelihood $\mathcal{L}_{\text{EnKF}}/\mathcal{L}_{\text{PF}}$ during training (lower right), as a function of training iterations. Throughout, the shaded area corresponds to ± 2 std over 5 repeated runs. (§3.5.2).

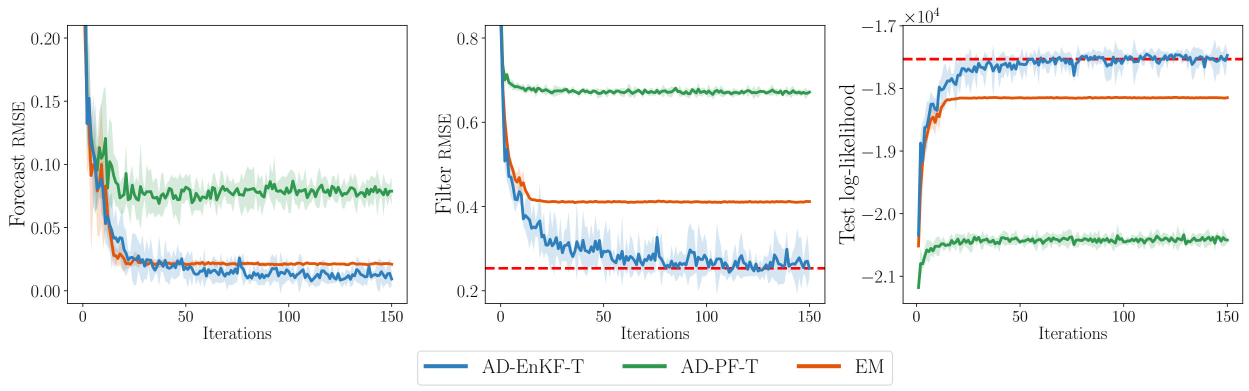


Figure 3.7: Learning parameterized dynamics of Lorenz-96 (3.5.3), with $d_x = 40$ and $H = I_{40}$. All performance metrics are evaluated after each training iteration. Red dashed lines correspond to metric values obtained with the reference model f^* and Q^*. Our proposed AD-EnKF-T performs the best in all metrics, with a performance similar to the reference model.

in partially-observed settings, so that the training process is not able to complete. Moreover, we observe that the error of AD-PF-T tends to grow with the state dimension d_x, while the two approaches based on EnKF do not deteriorate when increasing the state dimension. This is further evidence that EnKF is superior in high-dimensional settings.

	$d_x = 10$ (full)	$d_x = 20$ (full)	$d_x = 20$ (partial)	$d_x = 40$ (full)	$d_x = 40$ (partial)	$d_x = 80$ (full)	$d_x = 80$ (partial)
EM	0.308± 0.026	0.289± 0.0114	2.28± 4.92	0.268± 0.0103	7.754± 8.057	0.231± 0.0209	7.382± 4.812
AD-PF-T	0.262± 0.020	0.711± 0.0291	–	1.557± 0.0422	–	2.079± 0.0275	–
AD-EnKF-T	**0.217± 0.027**	**0.0325± 0.0128**	**0.0835± 0.0189**	**0.0283± 0.0022**	**0.0930± 0.0098**	**0.0540± 0.0065**	**0.0813± 0.0083**

Table 3.5.2: Lorenz-96, learning parameterized dynamics with varying d_x and observation models. The table shows recovery of learned α^* for each algorithm at the final training iteration, in terms of its distance to the truth α^* Eq. (3.5.4). "Full" corresponds to full observations, i.e., $H = I_{d_x}$. "Partial" corresponds to observing 2 out of 3 coordinates, i.e., $H = [e_1, e_2, e_4, e_5, e_7, \cdots]^\top$. The "-" indicates that training cannot be completed due to filter divergence. (§3.5.2).

Fully Unknown Dynamics

We assume no knowledge of the reference vector field f^*, and we approximate it by a neural network surrogate, $f_\alpha^{\mathrm{NN}} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, where here α represents the NN weights. The structure of the NN is similar to the one in [Brajard et al., 2020] and is detailed in §3.7.6. The number of parameters combined for α and β is $d_\theta = 9317$. The experimental results are compared to the model correction results, and hence are postponed to §3.5.2.

Model Correction

We assume f^* is unknown, but that an inaccurate model f_{approx} is available. We make use of the parametric form Eq. (3.5.3), and define f_{approx} via a perturbation $\widetilde{\alpha}$ of the true parameter α^*:

$$f_{\text{approx}} := f_{\widetilde{\alpha}}, \qquad \text{where } \widetilde{\alpha}_i \sim \begin{cases} \mathcal{N}(\alpha_i^*, 1), & \text{if } i = 0, \\ \mathcal{N}(\alpha_i^*, 0.1), & \text{if } i \in \{1, \dots, 5\}, \\ \mathcal{N}(\alpha_i^*, 0.01), & \text{if } i \in \{6, \dots, 17\}. \end{cases} \tag{3.5.5}$$

The coefficients of a higher order polynomial have a smaller amount of perturbation. $\widetilde{\alpha}$ is *fixed* throughout the learning procedure. We approximate the residual $f^* - f_{\text{approx}}$ by a NN g_α^{NN}, where α represents the weights, and g_α^{NN} has the same structure and the same number of parameters as in the fully unknown setting. The goal is to learn α so that $f_\alpha := f_{\text{approx}} + g_\alpha^{\text{NN}}$ approximates f^*.

We set $d_x = 40$ and consider two settings for H: fully observed with $H = I_{40}$, $d_y = 40$, and partially observed at every two out of three coordinates with $d_y = 27$ (see §3.5.2). Eight data sequences are generated with the reference model for training and four for testing, each with length $T = 1200$. Other experimental settings are the same as in §3.5.2.

For the setting where training data is fully observed, we compare AD-EnKF-T with AD-PF-T and the EM approach. The results are plotted in Fig. 3.8. The number of particles used for all algorithms is fixed at $N = 50$, and covariance tapering Eq. (3.7.45) with radius $r = 5$ is applied to EnKF. The subsequence length for both AD-EnKF-T and AD-PF-T is chosen to be $L = 20$. We find that, whether f^* is fully known or an inaccurate model is available, AD-EnKF-T is able to learn the reference vector field f^* well, with the smallest forecast RMSE among all methods. Applying a filtering algorithm to the learned model, we find that the states recovered by the AD-EnKF-T algorithm at the final iteration have the lowest error (filter RMSE) among all methods, indicating that AD-EnKF-T also has the ability to learn unknown states well. Moreover, the filter RMSE of AD-EnKF-T is close to the one computed using a filtering algorithm *with* known f^* and Q^*. The test log-likelihood $\mathcal{L}_{\text{EnKF}}$ of the model learned by AD-EnKF-T is close to the one evaluated with the reference model. We also find that having an inaccurate model f_{approx} is beneficial to the learning of AD-EnKF-T. The performance metrics are boosted compared to the ones with a fully unknown model, in agreement with [Levine and Stuart, 2021]. EM has worse results, where we find that the forecast RMSE does not consistently drop in the training procedure and the states are not accurately recovered. This might be because the smoothing distribution used by EM cannot be approximated accurately. AD-PF-T has the worst performance, possibly because PF fails in high dimensions.

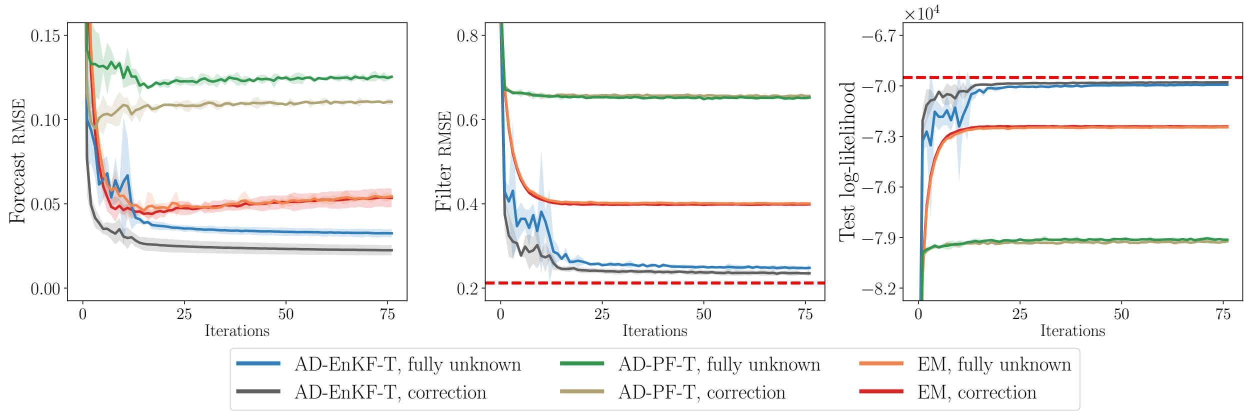


Figure 3.8: Learning the Lorenz-96 model from fully unknown dynamics (§3.5.2) v.s. model correction (§3.5.2), with full observations ($H = I_{d_x}$). All performance metrics are evaluated after each training iteration. Red dashed lines correspond to metric values obtained with the reference model f^* and Q^*. Our proposed AD-EnKF-T performs the best in all metrics, with a performance similar to the reference model.

We repeat the learning procedure in the setting where training data is partially observed at every two out of three coordinates. The results are shown in Fig. 3.9. Those for AD-PF-T are not shown since training cannot be completed due to filter divergence. We find that AD-EnKF-T is still able to recover f^* consistently as well as the unknown states for all coordinates, including the ones that are not observed, and has a filter RMSE close to the one computed with knowledge of f^*. However, the performance metrics of the EM algorithm in the model correction experiment deteriorate as training proceeds, indicating that it may overfit the training data. In addition, we find that the EM algorithm does not converge to the same point in repeated trials, particularly so in the setting of fully unknown dynamics. All of these results indicate that AD-EnKF is advantageous when learning from partial observations in high dimensions.

The ability to recover the underlying dynamics and states even with incomplete observations and fully unknown dynamics is most likely due to the convolutional-type architecture of the NN f_α^{NN}, which implicitly assumes that each coordinate only interacts with its neighbors, and that this interaction is spatially invariant.

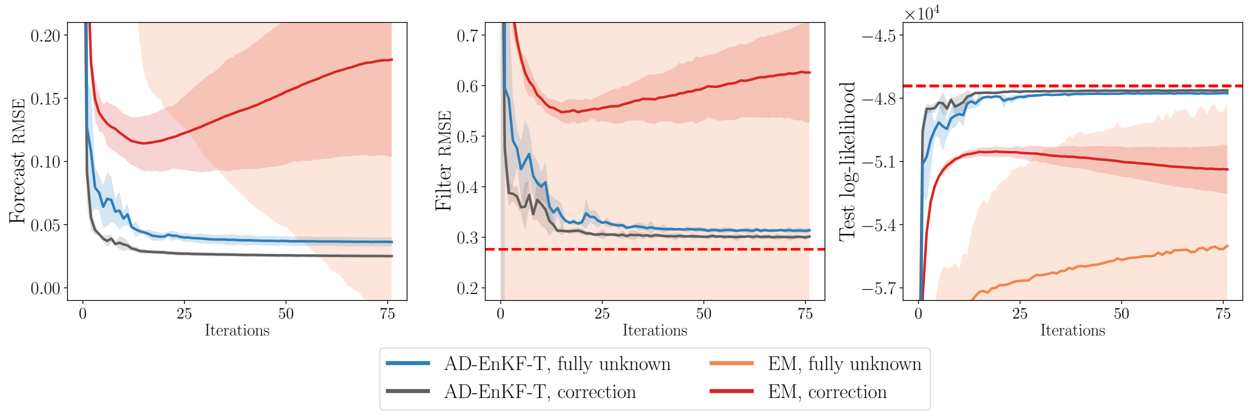


Figure 3.9: Learning Lorenz-96 from fully unknown dynamics (§3.5.2) v.s. model correction (§3.5.2) with partial observations ($H = [e_1, e_2, e_4, e_5, e_7, \cdots]^\top$). All performance metrics are evaluated after each training iteration. Red dashed lines correspond to metric values obtained with the reference model f^* and Q^*. The absence of lines for EM in the fully unknown setting is due to its low and unstable performance. When compared to the EM method, our proposed AD-EnKF-T is more stable during training, performs better in all metrics, and its performance is closer to the one achieved by the reference model.

3.6 Conclusions and Future Directions

This paper introduced AD-EnKFs for the principled learning of states and dynamics in DA. We have shown that AD-EnKFs can be successfully integrated with DA localization techniques for recovery of high-dimensional states, and with TBPTT techniques to handle large observation data and high-dimensional surrogate models. Numerical results on the Lorenz-96 model show that AD-EnKFs outperform existing EM and PF methods to merge DA and ML.

Several research directions stem from this work. First, gradient and Hessian information of $\mathcal{L}_{\text{EnKF}}$ obtained by autodiff can be utilized to design optimization schemes beyond the first-order approach we consider. Second, the convergence analysis of EnKF estimation of the log-likelihood and its gradient may be generalized to nonlinear settings. It would also be interesting to derive a dimension-dependent bound for the L^p estimation error and the bias $\left|\mathbb{E}\,\mathcal{L}_{\text{EnKF}} - \mathcal{L}\right|$. Third, the idea of AD-EnKF could be applied to auto-differentiate through other filtering algorithms, e.g. unscented Kalman filters, and in Bayesian inverse problems using iterative ensemble Kalman methods. The paper [Kloss et al., 2021] is an important first step in this direction. Replacing EnKF analysis steps with differentiable optimal transport maps [Corenflos et al., 2021] is also a promising future direction. Finally, the encouraging numerical results obtained on the Lorenz-96 model motivate the deployment and further investigation of AD-EnKFs in scientific and engineering applications where latent states need to be estimated with incomplete knowledge of their dynamics.

3.7 Appendix

3.7.1 Proof of Theorem 3.3.1

Notation We denote by c a constant that does not depend on N and may change from line to line. We denote by $\|U\|_p$ the L^p norm of a random vector/matrix U: $\|U\|_p := \left(\mathbb{E}\,|U|^p\right)^{1/p}$, where $|\cdot|$ is the underlying vector/matrix norm. (Here we use 2-norm for vectors and Frobenius norm for matrices.) For a sequence of random vectors/matrices U_N, we write

$$U_N \xrightarrow{1/2} U$$

if, for any $p \geq 1$, there exists a constant c such that

$$\|U_N - U\|_p \leq cN^{-1/2}, \qquad \forall N \geq 1.$$

For a scalar valued function $f(U)$ that takes a vector/matrix U as input, we denote by $\partial_U f$ the derivative of f w.r.t. U, which collects the derivative of f w.r.t. each entry of the vector/matrix U. When U is a vector, the notations $\partial_U f$ and $\nabla_U f$ are equivalent. For a vector/matrix valued function $U(a)$ that takes a scalar a as input, we denote by $\partial_a U$ the derivative of U w.r.t. a, which collects the derivative of each entry of the vector/matrix U w.r.t. a.

We first recall the propagation of chaos statement. Notice that in the EnKF algorithm Alg. 3.3.1, we compute $x_t^{1:N}$ sequentially, based on the forecast ensemble $\widehat{x}_t^{1:N}$ and its empirical mean and covariance $\widehat{m}_t, \widehat{C}_t$. We build "substitute particles" $\mathbf{x}_t^{1:N}$ in a similar fashion, except that at each step the population mean and covariance $\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t$ are used instead of their empirical versions. Starting from $\mathbf{x}_0^{1:N} = x_0^{1:N}$, the update rules of substitute particles are listed below, with a side-by-side comparison to the EnKF update rules:

$$\begin{array}{rl|rl}
\multicolumn{2}{c|}{\text{EnKF particles}} & \multicolumn{2}{c}{\text{Substitute particles}} \\
\hline
\widehat{x}_t^n & = F_\alpha(x_{t-1}^n) + \xi_t^n & \widehat{\mathbf{x}}_t^n & = F_\alpha(\mathbf{x}_{t-1}^n) + \xi_t^n \\
\widehat{m}_t & = \frac{1}{N}\sum_{n=1}^N \widehat{x}_t^n & \widehat{\mathbf{m}}_t & = \mathbb{E}\big[\widehat{\mathbf{x}}_t^n\big] \\
\widehat{C}_t & = \frac{1}{N-1}\sum_{n=1}^N (\widehat{x}_t^n - \widehat{m}_t)(\widehat{x}_t^n - \widehat{m}_t)^\top & \widehat{\mathbf{C}}_t & = \mathbb{E}\big[(\widehat{\mathbf{x}}_t^n - \widehat{\mathbf{m}}_t)(\widehat{\mathbf{x}}_t^n - \widehat{\mathbf{m}}_t)^\top\big] \\
\widehat{K}_t & = \widehat{C}_t H^\top (H\widehat{C}_t H^\top + R)^{-1} & \widehat{\mathbf{K}}_t & = \widehat{\mathbf{C}}_t H^\top (H\widehat{\mathbf{C}}_t H^\top + R)^{-1} \\
x_t^n & = \widehat{x}_t^n + \widehat{K}_t(y_t + \gamma_t^n - H\widehat{x}_t^n) & \mathbf{x}_t^n & = \widehat{\mathbf{x}}_t^n + \widehat{\mathbf{K}}_t(y_t + \gamma_t^n - H\widehat{\mathbf{x}}_t^n)
\end{array} \tag{3.7.1}$$

Notice that the substitute particles use the *same* realization of random variables as the EnKF particles, including initialization of particles $x_0^{1:N}$, forecast simulation error ξ_t^n, and noise perturbation γ_t^n. As $N \to \infty$, one can show that the EnKF particles $x_t^{1:N}$ (resp. $\widehat{x}_t^{1:N}$) are close to the substitute particles $\mathbf{x}_t^{1:N}$ (resp. $\widehat{\mathbf{x}}_t^{1:N}$), and hence the law of large numbers guarantees that $\widehat{m}_t, \widehat{C}_t$ are close to $\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t$. We summarize the main results from [Le Gland et al., 2009] (see also [Kwiatkowski and Mandel, 2015]):

Lemma 3.7.1. *Under the same assumption of Theorem 3.3.1:*

(1) For each $t \geq 1$, the substitute particles $\mathbf{x}_t^{1:N}$ are i.i.d., and each of them has the same law as the true filtering distribution $p(x_t|y_{1:t})$. Similarly, $\widehat{\mathbf{x}}_t^{1:N}$ are i.i.d., and each of them has the same law as the true forecast distribution $p(x_t|y_{1:t-1})$. In particular,

$$p(x_t|y_{1:t-1}) = \mathcal{N}(x_t; \widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t). \tag{3.7.2}$$

(2) For each $t, n, p \geq 1$, the EnKF particle x_t^n converges to the substitute particle $\mathbf{x}_t^n$ in L^p with convergence rate $N^{-1/2}$, and the substitute particle $\mathbf{x}_t^n$ has finite moments of any order. The same holds for forecast particles $\widehat{x}_t^n$:

$$x_t^n \xrightarrow{1/2} \mathbf{x}_t^n, \qquad \widehat{x}_t^n \xrightarrow{1/2} \widehat{\mathbf{x}}_t^n, \qquad \|\mathbf{x}_t^n\|_p \leq c, \qquad \|\widehat{\mathbf{x}}_t^n\|_p \leq c. \tag{3.7.3}$$

In particular, $\widehat{m}_t$, $\widehat{C}_t$ converge to $\widehat{\mathbf{m}}_t$, $\widehat{\mathbf{C}}_t$ in L^p with convergence rate $N^{-1/2}$:

$$\widehat{m}_t \xrightarrow{1/2} \widehat{\mathbf{m}}_t, \qquad \widehat{C}_t \xrightarrow{1/2} \widehat{\mathbf{C}}_t. \tag{3.7.4}$$

Proof. Eq. (3.7.2) corresponds to Lemma 2.1 of [Le Gland et al., 2009]. Eq. (3.7.3) corresponds to Proposition 4.4 of [Le Gland et al., 2009]. Eq. (3.7.4) is a direct corollary of Theorem 5.2 of [Le Gland et al., 2009]. □

Proof of Theorem 3.3.1. By Eq. (3.7.2), using the Gaussian observation model Eq. (3.2.2):

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \log p(y_t|y_{1:t-1}) = \sum_{t=1}^{T} \log \mathcal{N}(y_t; H\widehat{\mathbf{m}}_t, H\widehat{\mathbf{C}}_t H^\top + R). \tag{3.7.5}$$

By Eq. (3.3.6),

$$\mathcal{L}_{\text{EnKF}}(\theta) = \sum_{t=1}^{T} \log \mathcal{N}(y_t; H\widehat{m}_t, H\widehat{C}_t H^\top + R). \tag{3.7.6}$$

Define

$$\begin{aligned} h_t(m, C) &:= \log \mathcal{N}(y_t; Hm, HCH^\top + R) \\ &= -\frac{1}{2} \log \det(HCH^\top + R) - \frac{1}{2}(y_t - Hm)^\top (HCH^\top + R)^{-1}(y_t - Hm) + \text{const.} \end{aligned} \tag{3.7.7}$$

It suffices to show that, for each $t \geq 1$,

$$h_t(\widehat{m}_t, \widehat{C}_t) \xrightarrow{1/2} h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t). \tag{3.7.8}$$

We denote by $\mathbb{S}_+^{d_x} \subset \mathbb{R}^{d_x \times d_x}$ the space of all positive semi-definite matrices equipped with Frobenius norm. Notice that h_t is a continuous function on $\mathbb{R}^{d_x} \times \mathbb{S}_+^{d_x}$, since $HCH^\top + R \succeq R \succ 0$. To show convergence in L^p, intuitively one would expect a Lipschitz-type continuity to hold for h_t, in a suitable sense. We inspect the derivatives of h_t w.r.t. m and C, which will also be useful for later developments:

$$\begin{aligned} \partial_m h_t(m, C) &= -H^\top (HCH^\top + R)^{-1}(y_t - Hm), \\ \partial_C h_t(m, C) &= -\frac{1}{2} H^\top (HCH^\top + R)^{-1} H \\ &\qquad + \frac{1}{2} H^\top (HCH^\top + R)^{-1}(y_t - Hm)(y_t - Hm)^\top (HCH^\top + R)^{-1} H. \end{aligned} \tag{3.7.9}$$

Since $R^{d_x} \times \mathbb{S}_+^{d_x}$ is convex, by the mean value theorem, triangle inequality and Cauchy-Schwarz, and define $m(\chi) := \chi \widehat{\mathbf{m}}_t + (1 - \chi)\widehat{m}_t$, $C(\chi) := \chi \widehat{\mathbf{C}}_t + (1 - \chi)\widehat{C}_t$,

$$\begin{aligned} \left| h_t(\widehat{m}_t, \widehat{C}_t) - h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t) \right| \leq& \sup_{\chi \in [0,1]} \left| \partial_m h_t\big(m(\chi), C(\chi)\big) \right| |\widehat{m}_t - \widehat{\mathbf{m}}_t| \\ &+ \sup_{\chi \in [0,1]} \left| \partial_C h_t\big(m(\chi), C(\chi)\big) \right| |\widehat{C}_t - \widehat{\mathbf{C}}_t|. \end{aligned} \tag{3.7.10}$$

Taking L_p norm on both sides,

$$\begin{aligned}\big\|h_t(\widehat{m}_t, \widehat{C}_t) - h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t)\big\|_p \leq \sup_{\chi\in[0,1]} \big\|\partial_m h_t\big(m(\chi), C(\chi)\big)\big\|_{2p} \|\widehat{m}_t - \widehat{\mathbf{m}}_t\|_{2p} \\ + \sup_{\chi\in[0,1]} \big\|\partial_C h_t\big(m(\chi), C(\chi)\big)\big\|_{2p} \|\widehat{C}_t - \widehat{\mathbf{C}}_t\|_{2p},\end{aligned} \tag{3.7.11}$$

where we have used the triangle inequality and the L^p Cauchy-Schwarz inequality $\||U||V|\|_p \leq \|U\|_{2p}\|V\|_{2p}$, see e.g., Lemma 2.1 of [Kwiatkowski and Mandel, 2015]. Also, by plugging in *Eq.* (3.7.9), for each $\chi \in [0, 1]$,

$$\begin{aligned}\big\|\partial_m h_t\big(m(\chi), C(\chi)\big)\big\|_{2p} &\leq \big\||H||(HC(\chi)H^\top + R)^{-1}|(|y_t - \chi H\widehat{\mathbf{m}}_t - (1-\chi)H\widehat{m}_t|)\big\|_{2p} \\ &\leq |H||R^{-1}|\big(|y_t| + |H||\widehat{\mathbf{m}}_t| + |H|\|\widehat{m}_t\|_{2p}\big) \leq c,\end{aligned} \tag{3.7.12}$$

where we have used that $|(HC(\chi)H^\top + R)^{-1}| \leq |R^{-1}|$, that $\widehat{\mathbf{m}}_t$ is deterministic, and that all moments of $\widehat{m}_t$ are finite, by Eq. (3.7.4). Similarly,

$$\begin{aligned}\big\|\partial_C h_t\big(m(\chi), C(\chi)\big)\big\|_{2p} &\leq \frac{1}{2}|H|^2|R^{-1}| + \frac{1}{2}|H|^2|R^{-1}|^2\|y_t - \chi H\widehat{\mathbf{m}}_t - (1-\chi)H\widehat{m}_t\|_{4p}^2 \\ &\leq \frac{1}{2}|H|^2|R^{-1}| + \frac{1}{2}|H|^2|R^{-1}|^2(|y_t| + |H||\widehat{\mathbf{m}}_t| + |H|\|\widehat{m}_t\|_{4p})^2 \leq c,\end{aligned} \tag{3.7.13}$$

where we have used that $|vv^\top| = |v|^2$ for vector v. Thus, combining Eqs. (3.7.4) and (3.7.11) to (3.7.13) gives

$$\|h_t(\widehat{m}_t, \widehat{C}_t) - h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t)\|_p \leq cN^{-1/2}, \tag{3.7.14}$$

which concludes the proof. □

3.7.2 Proof of Theorem 3.3.2

Without loss of generality, we assume that $\theta \in \mathbb{R}$ is a scalar parameter, since in general the gradient w.r.t. θ is a collection of derivatives w.r.t. each element of θ. We will use the following lemma repeatedly:

Lemma 3.7.2. *For sequences of random vectors/matrices U_N, V_N:*
(1) If $U_N - V_N \xrightarrow{1/2} 0$ and $V_N \xrightarrow{1/2} V$, then

$$U_N \xrightarrow{1/2} V. \tag{3.7.15}$$

(2) If $U_N \xrightarrow{1/2} U$, $V_N \xrightarrow{1/2} V$ and U, V have finite moments of any order, then

$$U_N V_N \xrightarrow{1/2} UV. \tag{3.7.16}$$

More generally, the result holds for multiplication of more than two variables.

Proof. (1) Using $U_N - V = (U_N - V_N) + (V_N - V)$, the proof follows from the triangle inequality.
(2) Applying triangle inequality and L^p Cauchy-Schwarz inequality,

$$\begin{aligned} \|U_N V_N - UV\|_p &\le \|(U_N - U)V_N\|_p + \|U(V_N - V)\|_p \\ &\le \|U_N - U\|_{2p}\|V_N\|_{2p} + \|U\|_{2p}\|V_N - V\|_{2p} \\ &\le cN^{-1/2}\Big(\|V\|_{2p} + cN^{-1/2}\Big) + \|U\|_{2p}cN^{-1/2} \\ &\le cN^{-1/2}. \end{aligned} \tag{3.7.17}$$

□

The following result, which we will use repeatedly, is an immediate corollary of Theorem 3.7.1:

Lemma 3.7.3. *Under the same assumption of Theorem 3.3.1,*

$$(H\widehat{C}_tH^\top + R)^{-1} \xrightarrow{1/2} (H\widehat{\mathbf{C}}_tH^\top + R)^{-1}. \tag{3.7.18}$$

Proof. Using the identity $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$ for invertible matrices A, B:

$$\begin{aligned} &\|(H\widehat{C}_tH^\top + R)^{-1} - (H\widehat{\mathbf{C}}_tH^\top + R)^{-1}\|_p \\ &= \|(H\widehat{C}_tH^\top + R)^{-1}H(\widehat{\mathbf{C}}_t - \widehat{C}_t)H^\top(H\widehat{\mathbf{C}}_tH^\top + R)^{-1}\|_p \\ &\leq |R^{-1}|^2|H|^2\|\widehat{C}_t - \widehat{\mathbf{C}}_t\|_p \\ &\leq cN^{-1/2}, \end{aligned} \tag{3.7.19}$$

where we have used the L^p convergence of $\widehat{C}_t$ to $\widehat{\mathbf{C}}_t$ Eq. (3.7.4), and the fact that $|(HCH^T + R)^{-1}| \leq |R^{-1}|$ for $C \succeq 0$. □

Lemma 3.7.4. *Under the same assumption of Theorem 3.3.2, for each $t \geq 1$, both $\partial_\theta \widehat{x}_t^n$ and $\partial_\theta \widehat{\mathbf{x}}_t^n$ exist, and $\partial_\theta \widehat{x}_t^n$ converges to $\partial_\theta \widehat{\mathbf{x}}_t^n$ in L^p for any $p \geq 1$ with convergence rate $N^{-1/2}$. Moreover, $\partial_\theta \widehat{\mathbf{x}}_t^n$ has finite moments of any order:*

$$\partial_\theta \widehat{x}_t^n \xrightarrow{1/2} \partial_\theta \widehat{\mathbf{x}}_t^n, \qquad \|\partial_\theta \widehat{\mathbf{x}}_t^n\|_p \leq c, \qquad \forall n. \tag{3.7.20}$$

In addition, all derivatives $\partial_\theta \widehat{m}_t$, $\partial_\theta \widehat{\mathbf{m}}_t$, $\partial_\theta \widehat{C}_t$, $\partial_\theta \widehat{\mathbf{C}}_t$, $\partial_\theta \widehat{K}_t$ and $\partial_\theta \widehat{\mathbf{K}}_t$ exist, and

$$\partial_\theta \widehat{m}_t \xrightarrow{1/2} \partial_\theta \widehat{\mathbf{m}}_t, \qquad \partial_\theta \widehat{C}_t \xrightarrow{1/2} \partial_\theta \widehat{\mathbf{C}}_t, \qquad \partial_\theta \widehat{K}_t \xrightarrow{1/2} \partial_\theta \widehat{\mathbf{K}}_t. \tag{3.7.21}$$

Proof. We will prove this by induction. For $t = 1$, since $\widehat{x}_1^n = Ax_0^n + S\xi_0^n = \widehat{\mathbf{x}}_1^n$,

$$\partial_\theta \widehat{x}_1^n = (\partial_\theta A)x_0^n + (\partial_\theta S)\xi_0^n = \partial_\theta \widehat{\mathbf{x}}_1^n, \tag{3.7.22}$$

and both derivatives $\partial_\theta \widehat{x}_1^n$ and $\partial_\theta \widehat{\mathbf{x}}_1^n$ exist. Also,

$$\|\partial_\theta \widehat{\mathbf{x}}_1^n\|_p \leq |\partial_\theta A| \|x_0^n\|_p + |\partial_\theta S| \|\xi_0^n\|_p \leq c, \tag{3.7.23}$$

since x_0^n and ξ_0^n are drawn from Gaussian distributions, which have finite moments of any order. So Eq. (3.7.20) holds for $t = 1$.

Assume Eq. (3.7.20) holds for step t. Then, using the definition for $\widehat{m}_t$:

$$\partial_\theta \widehat{m}_t = \frac{1}{N} \sum_{n=1}^{N} \partial_\theta \widehat{x}_t^n \xrightarrow[①]{1/2} \frac{1}{N} \sum_{n=1}^{N} \partial_\theta \widehat{\mathbf{x}}_t^n \xrightarrow[②]{1/2} \mathbb{E}[\partial_\theta \widehat{\mathbf{x}}_t^n] \underset{③}{=} \partial_\theta \, \mathbb{E}[\widehat{x}_t^n] = \partial_\theta \widehat{\mathbf{m}}_t. \tag{3.7.24}$$

Convergence ① follows from induction assumption Eq. (3.7.20). Convergence ② follows from law of large numbers in L^p, since $\partial_\theta \widehat{\mathbf{x}}_t^n$ are i.i.d. and the moments of $\partial_\theta \widehat{\mathbf{x}}_t^n$ are finite by induction assumption Eq. (3.7.20). The swap of differentiation and expectation in ③ is valid since the expectation is taken over a distribution that is independent of θ. Both derivatives $\partial_\theta \widehat{m}_t$ and $\partial_\theta \widehat{\mathbf{m}}_t$ exist. Similarly,

$$\begin{aligned}
\partial_\theta \widehat{C}_t &= \frac{1}{N-1} \sum_{n=1}^{N} \partial_\theta (\widehat{x}_t^n (\widehat{x}_t^n)^\top) - \frac{N}{N-1} \partial_\theta (\widehat{m}_t \widehat{m}_t^\top) \\
&= \frac{1}{N-1} \sum_{n=1}^{N} \big((\partial_\theta \widehat{x}_t^n)(\widehat{x}_t^n)^\top + \widehat{x}_t^n (\partial_\theta \widehat{x}_t^n)^\top \big) - \frac{N}{N-1} \big((\partial_\theta \widehat{m}_t) \widehat{m}_t^\top + \widehat{m}_t (\partial_\theta \widehat{m}_t)^\top \big) \\
&\xrightarrow[①]{1/2} \frac{1}{N-1} \sum_{n=1}^{N} \big((\partial_\theta \widehat{\mathbf{x}}_t^n)(\widehat{\mathbf{x}}_t^n)^\top + \widehat{\mathbf{x}}_t^n (\partial_\theta \widehat{\mathbf{x}}_t^n)^\top \big) - \frac{N}{N-1} \big((\partial_\theta \widehat{\mathbf{m}}_t) \widehat{\mathbf{m}}_t^\top + \widehat{\mathbf{m}}_t (\partial_\theta \widehat{\mathbf{m}}_t)^\top \big) \\
&\xrightarrow[②]{1/2} \mathbb{E}[\partial_\theta \big(\widehat{\mathbf{x}}_t^n (\widehat{\mathbf{x}}_t^n)^\top \big)] - \partial_\theta (\widehat{\mathbf{m}}_t \widehat{\mathbf{m}}_t^\top) \\
&\underset{③}{=} \partial_\theta (\mathbb{E}[\widehat{\mathbf{x}}_t^n (\widehat{\mathbf{x}}_t^n)^\top] - \widehat{\mathbf{m}}_t \widehat{\mathbf{m}}_t^\top) \\
&= \partial_\theta \widehat{\mathbf{C}}_t.
\end{aligned} \tag{3.7.25}$$

For ① we have used the L^p convergence of $\widehat{x}_t^n$ to $\widehat{\mathbf{x}}_t^n$, $\partial_\theta \widehat{x}_t^n$ to $\partial_\theta \widehat{\mathbf{x}}_t^n$, $\widehat{m}_t$ to $\widehat{\mathbf{m}}_t$ and $\partial_\theta \widehat{m}_t$ to $\partial_\theta \widehat{\mathbf{m}}_t$ with rate $N^{-1/2}$, and the fact that $\widehat{\mathbf{x}}_t^n$ and $\partial_\theta \widehat{\mathbf{x}}_t^n$ have finite moments of any order, followed by Theorem 3.7.2. Convergence ② follows from law of large numbers in L^p since $(\partial_\theta \widehat{\mathbf{x}}_t^n)(\widehat{\mathbf{x}}_t^n)^\top$ are i.i.d. with finite moments, by the Cauchy-Schwarz inequality. ③ is valid since the expectation is taken over a distribution that is independent of θ. Both derivatives $\partial_\theta \widehat{C}_t$ and $\partial_\theta \widehat{\mathbf{C}}_t$ exist. Similarly,

$$
\begin{aligned}
\partial_\theta \widehat{K}_t &= \partial_\theta\big(\widehat{C}_t H (H\widehat{C}_t H^\top + R)^{-1}\big) \\
&\underset{①}{=} (\partial_\theta \widehat{C}_t) H (H\widehat{C}_t H^\top + R)^{-1} \\
&\quad - \widehat{C}_t H (H\widehat{C}_t H^\top + R)^{-1} (H(\partial_\theta \widehat{C}_t)H^\top + R)(H\widehat{C}_t H^\top + R)^{-1} \\
&\xrightarrow[②]{1/2} (\partial_\theta \widehat{\mathbf{C}}_t) H (H\widehat{\mathbf{C}}_t H^\top + R)^{-1} \qquad (3.7.26) \\
&\quad - \widehat{\mathbf{C}}_t H (H\widehat{\mathbf{C}}_t H^\top + R)^{-1} (H(\partial_\theta \widehat{\mathbf{C}}_t)H^\top + R)(H\widehat{\mathbf{C}}_t H^\top + R)^{-1} \\
&\underset{①}{=} \partial_\theta\big(\widehat{\mathbf{C}}_t H (H\widehat{\mathbf{C}}_t H^\top + R)^{-1}\big) \\
&= \partial_\theta \widehat{\mathbf{K}}_t.
\end{aligned}
$$

Here equalities ① and ③ follow from chain rule. For ② we have used the L^p convergence of $\widehat{C}_t$ to $\widehat{\mathbf{C}}_t$, $\partial_\theta \widehat{C}_t$ to $\partial_\theta \widehat{\mathbf{C}}_t$ and $(H\widehat{C}_t H^\top + R)^{-1}$ to $(H\widehat{\mathbf{C}}_t H^\top + R)^{-1}$ with rate $N^{-1/2}$ (by Eq. (3.7.18)), followed by Theorem 3.7.2. Both derivatives $\partial_\theta \widehat{K}_t$ and $\partial_\theta \widehat{\mathbf{K}}_t$ exist since $R \succ 0$.

To show Eq. (3.7.20) holds for step $t+1$, we need to investigate the derivatives of the analysis ensemble $\partial_\theta x_t^n$, by plugging in the EnKF update formula:

$$
\begin{aligned}
\partial_\theta x_t^n &= \partial_\theta(\widehat{x}_t^n + \widehat{K}_t(y_t + \gamma_t^n - H\widehat{x}_t^n) \\
&\underset{①}{=} (I - \widehat{K}_t H)\partial_\theta \widehat{x}_t^n + (\partial_\theta \widehat{K}_t)(y_t + \gamma_t^n - H\widehat{x}_t^n) \\
&= \big(I - \widehat{C}_t H^\top (H\widehat{C}_t H^\top + R)^{-1} H\big)\partial_\theta \widehat{x}_t^n + (\partial_\theta \widehat{K}_t)(y_t + \gamma_t^n - H\widehat{x}_t^n) \\
&\xrightarrow[②]{1/2} (I - \widehat{\mathbf{C}}_t H^\top (H\widehat{\mathbf{C}}_t H^\top + R)^{-1} H)\partial_\theta \widehat{\mathbf{x}}_t^n + (\partial_\theta \widehat{\mathbf{K}}_t)(y_t + \gamma_t^n - H\widehat{\mathbf{x}}_t^n) \qquad (3.7.27) \\
&= (I - \widehat{\mathbf{K}}_t H)\partial_\theta \widehat{\mathbf{x}}_t^n + (\partial_\theta \widehat{\mathbf{K}}_t)(y_t + \gamma_t^n - H\widehat{\mathbf{x}}_t^n) \\
&\underset{③}{=} \partial_\theta(\widehat{\mathbf{x}}_t^n + \widehat{\mathbf{K}}_t(y_t + \gamma_t^n - H\widehat{\mathbf{x}}_t^n) \\
&= \partial_\theta \mathbf{x}_t^n.
\end{aligned}
$$

Equalities ① and ③ follow from chain rule. For ② we have used the L^p convergence of $\widehat{x}_t^n$ to $\widehat{\mathbf{x}}_t^n$, $\partial_\theta \widehat{x}_t^n$ to $\partial_\theta \widehat{\mathbf{x}}_t^n$, $\widehat{C}_t$ to $\widehat{\mathbf{C}}_t$, $\partial_\theta \widehat{K}_t$ to $\partial_\theta \widehat{\mathbf{K}}_t$, and $(H\widehat{C}_t H^\top + R)^{-1}$ to $(H\widehat{\mathbf{C}}_t H^\top + R)^{-1}$, with convergence rate $N^{-1/2}$, and the fact that $\widehat{\mathbf{x}}_t^n$, $\partial_\theta \widehat{\mathbf{x}}_t^n$ and the Gaussian random variable γ_t^n have finite moments of any order, followed by Theorem 3.7.2. Both derivatives $\partial_\theta x_t^n$ and $\partial_\theta \mathbf{x}_t^n$ exist since $R \succ 0$. We also have the moment bound on $\partial_\theta \mathbf{x}_t^n$:

$$
\|\partial_\theta \mathbf{x}_t^n\| \le |I - \widehat{\mathbf{K}}_t H| \|\partial_\theta \widehat{\mathbf{x}}_t^n\|_p + |\partial_\theta \widehat{\mathbf{K}}_t|(|y_t| + \|\gamma_t^n\|_p + |H| \|\widehat{\mathbf{x}}_t^n\|_p) \le c, \qquad (3.7.28)
$$

since $\widehat{\mathbf{x}}_t^n$, $\partial_\theta \widehat{\mathbf{x}}_t^n$ and the Gaussian random variable γ_t^n have finite moments of any order.

Then,

$$\begin{aligned}
\partial_\theta \widehat{x}_{t+1}^n &= \partial_\theta(Ax_t^n + S\xi_t^n) \\
&\underset{①}{=} (\partial_\theta A)x_t^n + A(\partial_\theta x_t^n) + (\partial_\theta S)\xi_t^n \\
&\xrightarrow[②]{1/2} (\partial_\theta A)\mathbf{x}_t^n + A(\partial_\theta \mathbf{x}_t^n) + (\partial_\theta S)\xi_t^n \\
&\underset{③}{=} \partial_\theta(A\mathbf{x}_t^n + S\xi_t^n) \\
&= \partial_\theta \widehat{\mathbf{x}}_{t+1}^n.
\end{aligned} \tag{3.7.29}$$

Here equalities ① and ③ follow from chain rule. For ② we have used the L^p convergence of x_t^n to $\mathbf{x}_t^n$ and $\partial_\theta x_t^n$ to $\partial_\theta \mathbf{x}_t^n$. Both derivatives $\partial_\theta \widehat{x}_{t+1}^n$ and $\partial_\theta \widehat{\mathbf{x}}_{t+1}^n$ exist since both $\partial_\theta x_t^n$ and $\partial_\theta \mathbf{x}_t^n$ exist. We also have the moment bound:

$$\|\partial_\theta \widehat{\mathbf{x}}_{t+1}^n\|_p \le |\partial_\theta A|\|\mathbf{x}_t^n\|_p + |A|\|\partial_\theta \mathbf{x}_t^n\|_p + |\partial_\theta S|\|\xi_t^n\|_p \le c, \tag{3.7.30}$$

since $\mathbf{x}_t^n$, $\partial_\theta \mathbf{x}_t^n$ and Gaussian random variable ξ_t^n have finite moments of any order. Thus Eq. (3.7.20) is proved for step $t+1$ and the induction step is finished, which concludes the proof of the lemma. □

Proof of Theorem 3.3.2. Recall the definition of h_t Eq. (3.7.7). It suffices to show that, for each $t \ge 1$,

$$\partial_\theta\Big(h_t(\widehat{m}_t, \widehat{C}_t)\Big) \xrightarrow{1/2} \partial_\theta\Big(h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t)\Big). \tag{3.7.31}$$

We first investigate the convergence of derivatives of h_t w.r.t $\widehat{m}_t$ and $\widehat{C}_t$. The derivatives are computed in Eq. (3.7.9):

$$\begin{aligned}
\partial_m h_t(\widehat{m}_t, \widehat{C}_t) &= -H^\top (H\widehat{C}_t H^\top + R)^{-1}(y_t - H\widehat{m}_t) \\
&\xrightarrow{1/2} -H^\top (H\widehat{\mathbf{C}}_t H^\top + R)^{-1}(y_t - H\widehat{\mathbf{m}}_t) \\
&= \partial_m h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t),
\end{aligned} \tag{3.7.32}$$

and

$$\begin{aligned}
\partial_C h_t(\widehat{m}_t, \widehat{C}_t) &= -\frac{1}{2} H^\top (H\widehat{C}_t H^\top + R)^{-1} H \\
&\quad + \frac{1}{2} H^\top (H\widehat{C}_t H^\top + R)^{-1} (y_t - H\widehat{m}_t)(y_t - H\widehat{m}_t)^\top (H\widehat{C}_t H^\top + R)^{-1} H \\
&\xrightarrow{1/2} -\frac{1}{2} H^\top (H\widehat{\mathbf{C}}_t H^\top + R)^{-1} H \\
&\quad + \frac{1}{2} H^\top (H\widehat{\mathbf{C}}_t H^\top + R)^{-1} (y_t - H\widehat{\mathbf{m}}_t)(y_t - H\widehat{\mathbf{m}}_t)^\top (H\widehat{\mathbf{C}}_t H^\top + R)^{-1} H,
\end{aligned} \tag{3.7.33}$$

where we have used the L^p convergence of $\widehat{m}_t$ to $\widehat{\mathbf{m}}_t$ and $(H\widehat{C}_t H^\top + R)^{-1}$ to $(H\widehat{\mathbf{C}}_t H^\top + R)^{-1}$ by Eq. (3.7.18), followed by Theorem 3.7.2. Then, by chain rule,

$$\begin{aligned}
\partial_\theta \Big(h_t(\widehat{m}_t, \widehat{C}_t) \Big) &= \Big(\partial_m h_t(\widehat{m}_t, \widehat{C}_t) \Big)^\top \partial_\theta \widehat{m}_t + \mathrm{Tr}\bigg(\Big(\partial_C h_t(\widehat{m}_t, \widehat{C}_t) \Big)^\top \partial_\theta \widehat{C}_t \bigg) \\
&\xrightarrow{1/2} \Big(\partial_m h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t) \Big)^\top \partial_\theta \widehat{\mathbf{m}}_t + \mathrm{Tr}\bigg(\Big(\partial_C h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t) \Big)^\top \partial_\theta \widehat{\mathbf{C}}_t \bigg) \\
&= \partial_\theta \Big(h_t(\widehat{\mathbf{m}}_t, \widehat{\mathbf{C}}_t) \Big).
\end{aligned} \tag{3.7.34}$$

Both derivatives exist since $\partial_\theta \widehat{m}_t$, $\partial_\theta \widehat{\mathbf{m}}_t$, $\partial_\theta \widehat{C}_t$ and $\partial_\theta \widehat{\mathbf{C}}_t$ exist, by Theorem 3.7.4. We have used Eq. (3.7.32) and Eq. (3.7.33) above, the L^p convergence of $\partial_\theta \widehat{m}_t$ to $\partial_\theta \widehat{\mathbf{m}}_t$ and $\partial_\theta \widehat{C}_t$ to $\partial_\theta \widehat{\mathbf{C}}_t$ with rate $N^{-1/2}$ by Theorem 3.7.4, followed by Theorem 3.7.2. □

Remark 3.7.5. *We again emphasize that all the derivatives and chain rule formulas do* not *need to be computed by hand in applications, but rather through the modern autodiff libraries. We list them out only for the purpose of proving convergence results.*

3.7.3 Additional Figures

See Fig. 3.10 and 3.11 for additional figures to the linear Gaussian experiment §3.5.1:

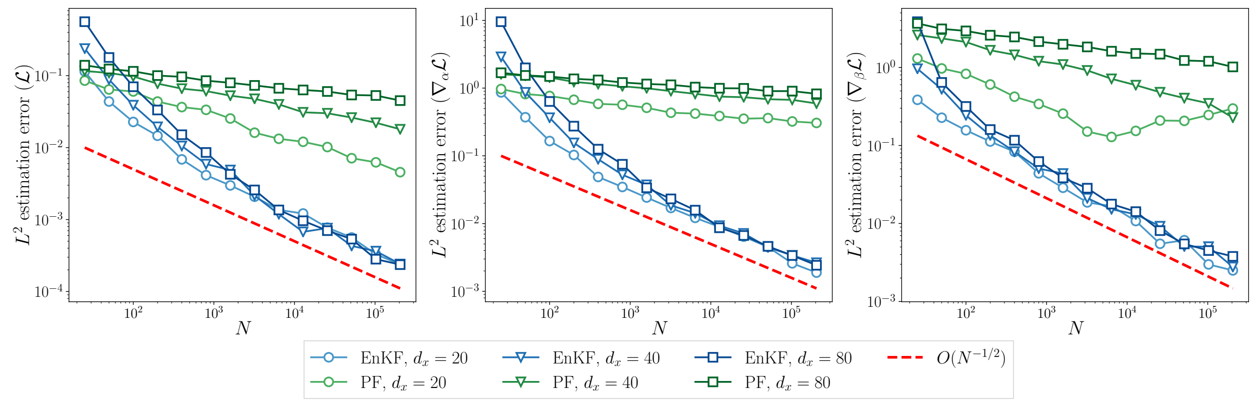


Figure 3.10: Relative L^2 estimation errors of the log-likelihood (left) and its gradient w.r.t. α (middle) and β (right), computed using EnKF and PF, as a function of N, for the linear-Gaussian model (3.5.1). State dimension $d_x \in \{20, 40, 80\}$. θ is evaluated at $\alpha = (0.5, 0.5, 0.5), \beta = (1, 0.1)$. (§3.5.1)

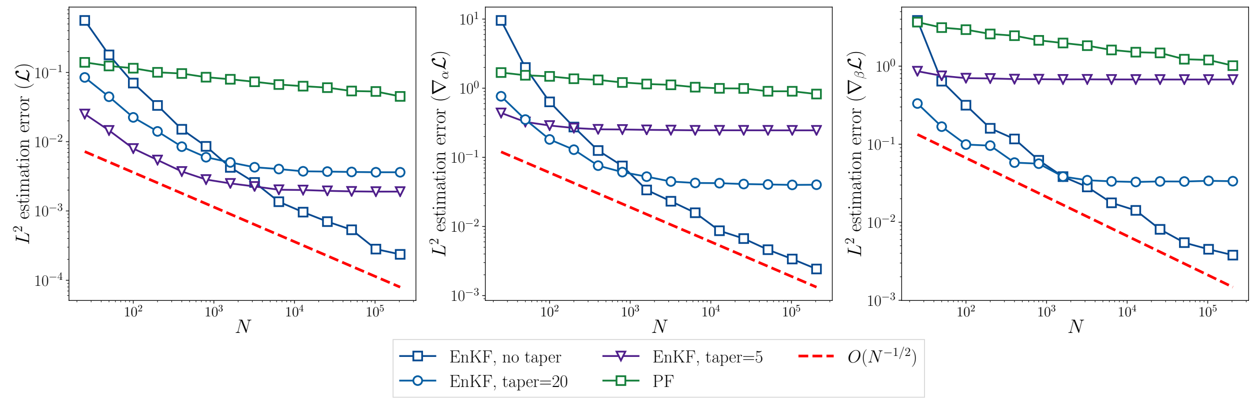


Figure 3.11: Relative L^2 estimation errors of log-likelihood (left) and its gradient w.r.t. α (middle) and β (right), computed using EnKF and PF, with different covariance tapering radius applied to EnKF. State dimension $d_x = 80$. θ is evaluated at $\alpha = (0.5, 0.5, 0.5), \beta = (1, 0.1)$.

3.7.4 Auto-differentiable Particle Filters

Here we review the auto-differentiable particle filter methods introduced in, e.g. [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018]. A particle filter (PF) propagates N weighted particles $(w_t^{1:N}, x_t^{1:N})$ to approximate the filtering distribution $p_\theta(x_t|y_{1:t})$. It first specifies a proposal distribution $r(x_t|x_{t-1}, y_t)$. Then it iteratively samples new particles from the proposal, followed by a reweighing procedure:

$$\widehat{x}_t^n \sim r(\cdot|x_{t-1}^n, y_t), \qquad \widehat{w}_t^n = \frac{p_\theta(\widehat{x}_t^n|x_{t-1}^n)\mathcal{N}(y_t; H\widehat{x}_t^n, R)}{r(\widehat{x}_t^n|x_{t-1}^n, y_t)} w_{t-1}^n. \tag{3.7.35}$$

A resampling step is performed when necessary, e.g., if the effective sample size drops below a certain threshold:

$$(w_t^n, x_t^n) = \Big(\frac{1}{N}, \widehat{x}_t^{a_t^n}\Big), \qquad \text{where } a_t^n \sim \text{Categorical}\big(\cdot|\widehat{w}_t^{1:N}\big). \tag{3.7.36}$$

We refer the reader to [Doucet and Johansen, 2009] for a more general introduction to PF. Here we consider PF with the optimal proposal $r(\widehat{x}_t^n|x_{t-1}^n, y_t) := p_\theta(\widehat{x}_t^n|x_{t-1}^n, y_t)$, as it is readily available for the family of SSMs that is considered in this paper [Doucet et al., 2000; Sanz-Alonso et al., 2018]. The unnormalized weights $\widehat{w}_t^n$ also provides an approximation to the data log-likelihood. Define

$$\mathcal{L}_{\text{PF}}(\theta) = \sum_{t=1}^T \log\Big(\sum_{n=1}^N \widehat{w}_t^n\Big). \tag{3.7.37}$$

Note that the weights $\widehat{w}_t^n$ depends implicitly on θ. It is a well-known result [Del Moral, 2004; Andrieu et al., 2010] that the data likelihood $\exp\mathcal{L}(\theta)$ can be unbiasedly estimated, and a lower bound for $\mathcal{L}(\theta)$ can be derived:

$$\mathbb{E}[\exp\mathcal{L}_{\text{PF}}(\theta)] = \exp\mathcal{L}(\theta), \qquad \mathbb{E}[\mathcal{L}_{\text{PF}}(\theta)] \leq \mathcal{L}(\theta), \tag{3.7.38}$$

where the last inequality follows from Jensen's inequality. We summarize the optimal PF algorithm and parameter learning procedure (AD-PF) in Alg. 3.7.1 and Alg. 3.7.2 below. AD-PF with truncated backprop (AD-PF-T) can be built in a similar fashion as AD-EnKF-T (Alg. 3.4.2), and is omitted. For a derivation of update rules of optimal PF, see e.g.

[Sanz-Alonso et al., 2018]. For AD-PF where the proposal is learned from data, see e.g. [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018]. Following the same authors, the gradients from the resampling steps are discarded. Therefore, $\nabla_\theta \mathcal{L}_{\mathrm{PF}}$ is biased and inconsistent as $N \to \infty$ (see [Corenflos et al., 2021]).

Algorithm 3.7.1 Optimal Particle Filtering and Log-likelihood Estimation

```
   **Input**: $\theta = \{\alpha, \beta\}, y_{1:T}, (w_0^{1:N}, x_0^{1:N})$. (If $(w_0^{1:N}, x_0^{1:N})$ is not specified, $x_0^n \overset{\text{i.i.d.}}{\sim} p_0(x_0)$, $w_0^n = 1/N$.)
1: **Initialize** $\mathcal{L}_{\mathrm{PF}}(\theta) = 0$.
2: Set $S = HQ_\beta H^\top + R$ and $K = Q_\beta H^\top S^{-1}$.
3: **for** $t = 1, \ldots, T$ **do**
4:     Set $\widehat{x}_t^n = (I - KH)F_\alpha(x_{t-1}^n) + Ky_t + \xi_t^n$, where $\xi_t^n \sim \mathcal{N}(0, Q_\beta)$.
5:     Set $\widehat{w}_t^n = \mathcal{N}\big(y_t; HF_\alpha(x_{t-1}^n), S\big)$
6:     Set $(w_t^n, x_t^n) = \Big(\frac{1}{N}, \widehat{x}_t^{a_t^n}\Big)$, where $a_t^n \sim \text{Categorical}\big(\cdot|\widehat{w}_t^{1:N}\big)$    ▷ Resampling step
7:     Set $\mathcal{L}_{\mathrm{PF}}(\theta) \leftarrow \mathcal{L}_{\mathrm{PF}}(\theta) + \log\Big(\sum_{n=1}^N \widehat{w}_t^n\Big)$.
8: **end for**
   **Output**: PF particles $(w_{0:T}^{1:N}, x_{0:T}^{1:N})$. Log-likelihood estimate $\mathcal{L}_{\mathrm{PF}}(\theta)$.
```

Algorithm 3.7.2 Auto-differentiable Particle Filtering (AD-PF)

```
   **Input**: Observations $y_{1:T}$. Learning rate $\eta$.
1: **Initialize** SSM parameter $\theta^0$ and set $k = 0$.
2: **while** not converging **do**
3:     $(w_{0:T}^{1:N}, x_{0:T}^{1:N}), \mathcal{L}_{\mathrm{PF}}(\theta^k) = \text{PARTICLEFILTER}(\theta^k, y_{1:T})$.    ▷ Alg. 3.7.1
4:     Compute $\nabla_\theta \mathcal{L}_{\mathrm{PF}}(\theta^k)$ by auto-differentiating the map $\theta^k \mapsto \mathcal{L}_{\mathrm{PF}}(\theta^k)$.
5:     Set $\theta^{k+1} = \theta^k + \eta \nabla_\theta \mathcal{L}_{\mathrm{PF}}(\theta^k)$ and $k \leftarrow k + 1$.
6: **end while**
   **Output**: Learned SSM parameter $\theta^k$ and PF particles $(w_{0:T}^{1:N}, x_{0:T}^{1:N})$.
```

3.7.5 Expectation-Maximization with Ensemble Kalman Filters

Here we review the EM methods introduced in, e.g., [Pulido et al., 2018; Bocquet et al., 2020]. Consider the log-likelihood expression

$$\begin{aligned}
\log p_\theta(y_{1:T}) &= \log \int p_\theta(y_{1:T}, x_{0:T}) \mathrm{d}x_{0:T} && (3.7.39)\\
&= \log \mathbb{E}_{q(x_{0:T}|y_{1:T})}\left[\frac{p_\theta(y_{1:T}, x_{0:T})}{q(x_{0:T}|y_{1:T})}\right] && (3.7.40)\\
&\geq \mathbb{E}_{q(x_{0:T}|y_{1:T})}\left[\log p_\theta(y_{1:T}, x_{0:T}) - \log q(x_{0:T}|y_{1:T})\right] := \mathcal{L}(q,\theta). && (3.7.41)
\end{aligned}$$

The algorithm maximizes $\mathcal{L}(q,\theta)$, which alternates between two steps:

1. (E-step) Fix θ^k. Find q^k that maximizes $\mathcal{L}(q,\theta^k)$, which can be shown to be the exact posterior

$$q^k(x_{0:T}|y_{1:T}) = p_{\theta^k}(x_{0:T}|y_{1:T}).$$

2. (M-step) Fix q^k. Find θ^{k+1} that maximizes $\mathcal{L}(q^k,\theta)$:

$$\theta^* = \arg\max_\theta \mathbb{E}_{q^k(x_{0:T}|y_{1:T})}\left[\log p_\theta(y_{1:T}, x_{0:T})\right]. \qquad (3.7.42)$$

The exact posterior in the E-step is not always available, so one can run an Ensemble Kalman Smoother (EnKS) to approximate the distribution $p_{\theta^k}(x_{0:T}|y_{1:T})$ by a group of equally-weighted particles $\{x_{0:T}^{k,n}\}_{n=1}^N$, and replace the expectation in the M-step by an Monte-Carlo average among these particles.

Consider again the nonlinear SSM defined by equations (3.2.1)-(3.2.2)-(3.2.3) with dynamics F_α and noise covariance Q_β. The key to the methods proposed in, e.g., [Pulido et al., 2018; Bocquet et al., 2020] is that, for the optimization problem Eq. (3.7.42) in the M-steps, optimizing w.r.t. α is hard, but optimizing w.r.t. β is easy. If we write $\{x_{0:T}^n\}_{n=1}^N$ as an approximation of $q(x_{0:T}|y_{1:T})$ (we drop k here for notation convenience), the optimization problem Eq. (3.7.42) can be rewritten as (using Monte-Carlo approximation)

$$\arg\max_{\theta} \frac{1}{N}\sum_{n=1}^{N} \log p_\theta(y_{1:T}, x_{0:T}^n) = \arg\max_{\alpha,\beta} \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T} \log \mathcal{N}\big(x_t^n; F_\alpha(x_{t-1}^n), Q_\beta\big).$$

When α is given, optimizing w.r.t. β becomes the usual MLE computation for covariance matrix of a multivariate Gaussian. We summarize the EM algorithm in Algorithm 3.7.3. Note that this special type of M-step update only applies to SSMs with transition kernel of the form Eq. (3.2.1). For more general SSMs, updates for β may not admit a closed form expression, and one may have to resort to a joint optimzation of (α, β) in the M-steps.

Algorithm 3.7.3 Expectation Maximization (EM) with Ensemble Kalman Filter

```
    Input: Observations $y_{1:T}$. Learning rate $\eta$. Smoothing parameter $S$. Inner-loop gradient ascent steps $J$.
 1: Initialize SSM parameter $\theta^0 = (\alpha^0, \beta^0)$. Initialize $k = 0$.
 2: while not converging do
 3:     $x_{0:T}^{1:N} = \text{EnsembleKalmanFilter}(\theta^k, y_{1:T})$.                ▷ Alg. 3.3.1
 4:     Perform smoothing for lag $S$ (see e.g. [Katzfuss et al., 2020]) and update $x_{0:T}^{1:N}$.
 5:     Set
            $$Q_{\beta^{k+1}} = \frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\Big(x_t^n - F_{\alpha^k}(x_{t-1}^n)\Big)\Big(x_t^n - F_{\alpha^k}(x_{t-1}^n)\Big)^\top.$$
 6:     Initialize $\alpha^{k,0} = \alpha^k$.
 7:     for $j = 0, \dots, J-1$ do
 8:         $\alpha^{k,j+1} = \alpha^{k,j} + \eta\nabla_\alpha \mathcal{L}_{\text{EM-EnKF}}(\alpha^{k,j}, x_{0:T}^{1:N})$, where
            $$\mathcal{L}_{\text{EM-EnKF}}(\alpha, x_{0:T}^{1:N}) := \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\log\mathcal{N}(x_t^n; F_\alpha(x_{t-1}^n), Q_{\beta^{k+1}}).$$
 9:     end for
10:     Set $\alpha^{k+1} = \alpha^{k,J}$, $k \leftarrow k+1$.
11: end while
    Output: Learned SSM parameter $\theta^k = (\alpha^k, \beta^k)$ and particles $x_{0:T}^{1:N}$.
```

3.7.6 Implementation Details and Additional Performance Metrics

Linear-Gaussian Model

The relative L^2 errors of the log-likelihood and gradient estimates in §3.5.1 are given by

$$\frac{\mathbb{E}[(\mathcal{L}_{\mathrm{EnKF}}(\theta)-\mathcal{L}(\theta))^2]^{1/2}}{|\mathcal{L}(\theta)|}, \tag{3.7.43}$$

and

$$\frac{\mathbb{E}[|\nabla_\alpha\mathcal{L}_{\mathrm{EnKF}}(\theta)-\nabla_\alpha\mathcal{L}(\theta)|^2]^{1/2}}{|\nabla_\alpha\mathcal{L}(\theta)|}, \qquad \frac{\mathbb{E}[|\nabla_\beta\mathcal{L}_{\mathrm{EnKF}}(\theta)-\nabla_\beta\mathcal{L}(\theta)|^2]^{1/2}}{|\nabla_\beta\mathcal{L}(\theta)|}. \tag{3.7.44}$$

The tapering matrix ρ in §3.5.1 is defined through $[\rho]_{i,j}=\varphi\Big(\frac{|i-j|}{r}\Big)$, where

$$\varphi(z)=\begin{cases} 1-\frac{5}{3}z^2+\frac{5}{8}z^3+\frac{1}{2}z^4-\frac{1}{4}z^5, & \text{if } 0\le z<1,\\ 4-5z+\frac{5}{3}z^2+\frac{5}{8}z^3-\frac{1}{2}z^4+\frac{1}{12}z^5-\frac{2}{3z}, & \text{if } 1\le z<2,\\ 0, & \text{if } z\ge 2. \end{cases} \tag{3.7.45}$$

We initialize the parameter learning at $\alpha^0=(0.5,0.5,0.5)$ and $\beta^0=(1,0.1)$. For a fair comparison of convergence performance, we set the learning rate to be $\eta_\alpha=10^{-4}$ for α and $\eta_\beta=10^{-3}$ for β for all methods, and perform gradient ascent for 1000 iterations.

Lorenz-96

Additional Performance Metrics

Averaged Diagnosed Error Level: Measures the level of the learned model error:

$$\sigma_\beta=\sqrt{\mathrm{Tr}(Q_\beta)/d_x}. \tag{3.7.46}$$

Forecast RMSE (RMSE-f): Measures the forecast error of the learned vector field f_α compared to the reference vector field f^* . We select $P = 4000$ points $\{x_p\}_{p=1}^P$ uniformly at random from the attractor of the reference system, by simulating a long independent run. The forecast RMSE is defined as

$$\text{RMSE-f} = \sqrt{\frac{1}{d_x P}\sum_{p=1}^{P}\Big|F_\alpha(x_p) - F^*(x_p)\Big|^2}. \tag{3.7.47}$$

Here we recall that F^* and F_α are the Δ_s-flow maps with fields f^* and f_α, where Δ_s is the time between observations.

Analysis/Filter RMSE (RMSE-a): Measures the state estimation error of the learned state transition kernel $p_\theta(x_t|x_{t-1})$. Let $(x_{0:T}^{\text{train}}, y_{1:T}^{\text{train}})$ be one sequence of training data. Then a filtering run is performed over the data $y_{1:T}^{\text{train}}$ with the learned state transition kernel $p_\theta(x_t|x_{t-1})$, using EnKF/PF, depending on which method is used for training. The (weighted) ensemble/particle mean $\bar{x}_{1:T}$ is recorded and the filter/analysis RMSE is defined as

$$\text{RMSE-a} = \sqrt{\frac{1}{d_x(T - T_b)}\sum_{t=T_b}^{T}|\bar{x}_t - x_t^{\text{train}}|^2}. \tag{3.7.48}$$

Here T_b is the number of burn-in steps. For simplicity, we set $T_b = \lfloor T/5 \rfloor$. If multiple sequences of training data are used, then the quantity inside the square root is further averaged among all training sequences.

Test Log-likelihood: Measures the approximate data log-likelihood under the learned state transition kernel. Let $(x_{0:T}^{\text{test}}, y_{1:T}^{\text{test}})$ be a sequence of test data drawn from the reference model independent of the training data used to estimate θ. The test log-likelihood is defined as $\mathcal{L}_{\text{EnKF}}(\theta)$ (or $\mathcal{L}_{\text{PF}}(\theta)$) computed over $y_{1:T}^{\text{test}}$ using the corresponding EnKF/PF approximation, depending on which method is used for training.

Implementation Details We use Adam [Kingma and Ba, 2014] throughout this experiment, with a learning rate that decays polynomially with the number of training iterations. Specifically, let i be the index of current iteration. The learning rate η_i is defined as:

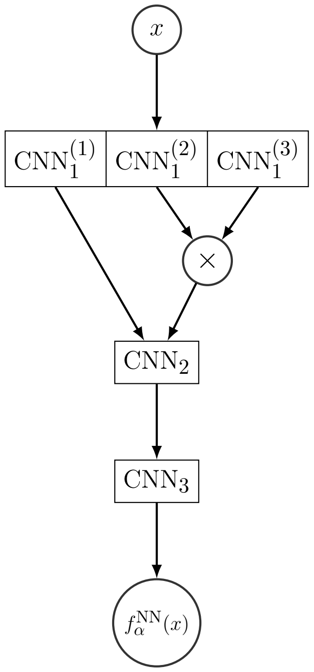
$$\eta_i = \begin{cases} \eta_0, & i \leq I_0, \\ \eta_0 (i - I_0)^{-\tau}, & i > I_0, \end{cases} \tag{3.7.49}$$

where η_0, I_0, and τ are hyperparaters to be specified. We set $I_0 = 10$ throughtout the experiment. β is initialized at $\beta^0 = 2 \cdot \mathbf{1}$ where $\mathbf{1} \in \mathbb{R}^{d_x}$ is the all-ones vector.

Parameterized Dynamics α is initialized at $\alpha^0 = \mathbf{0} \in \mathbb{R}^{18}$. For AD-EnKF-T and AD-PF-T we set $\eta_0 = 10^{-1}$ and $\tau = 0.5$. For EM we set $\eta_0 = 10^{-1}$, $\tau = 1$, inner-loop gradient ascent steps $J = 3$ and smoothing parameter $S = 0$. The choice of S is selected from $\{0, 2, 4, 6, 8\}$, following [Bocquet et al., 2020; Brajard et al., 2020]. Although EM-based approaches in theory require to compute the smoothing rather than filtering distribution (see e.g. [Bishop, 2006; Bocquet et al., 2020]), in practice the ensemble-based smoothing algorithms may fail to produce an accurate approximation. All hyperparameters are tuned on validation sets. We also find that only the choice of η_0 is crucial to the convergence of AD-EnKF-T empirically, while other hyperparameters (I_0 and τ) have small effects on the convergence speed.

Fully Unknown Dynamics NN weights α^0 are initialized at random using PyTorch's default initialization. For AD-EnKF-T, AD-PF-T and EM we set $\eta_0 = 10^{-2}$ and $\tau = 1$. For EM we set inner-loop gradient ascent steps $J = 3$ and smoothing parameter $S = 0$. The structure of f_α^{NN} is detailed in Fig. 3.12.

Model Correction NN weights α^0 are initialized at random using PyTorch's default initialization. For AD-EnKF-T and AD-PF-T we set $\eta_0 = 10^{-3}$ and $\tau = 0.75$. For EM we set $\eta_0 = 10^{-3}$ and $\tau = 1$, $J = 3$ and $S = 0$. The structure of g_α^{NN} is the same as f_α^{NN} in the previous experiment, see Fig. 3.12.



(a) Network structure

	#in	#out	kernel size
CNN_1	1	72(24×3)	5, circular
CNN_2	48(24×2)	37	5, circular
CNN_3	37	1	1

(b) Network details

Figure 3.12: Structure of f_α^{NN}. Output channels of CNN_1 is divided into three groups of equal length $\mathrm{CNN}_1^{(1)}$, $\mathrm{CNN}_1^{(2)}$ and $\mathrm{CNN}_1^{(3)}$. Input channels to CNN_2 is a concatenation of $\mathrm{CNN}_1^{(1)}$ and $(\mathrm{CNN}_1^{(2)} \times \mathrm{CNN}_1^{(3)})$, where the multiplication is point-wise.

CHAPTER 4
REDUCED-ORDER AUTODIFFERENTIABLE ENSEMBLE KALMAN FILTERS

4.1 Introduction

Reconstructing and forecasting a time-evolving state given partial and noisy time-series data is a fundamental problem in science and engineering, with far-ranging applications in numerical weather forecasting, climate, econometrics, signal processing, stochastic control, and beyond. Two common challenges are the presence of *model error* in the dynamics governing the evolution of the state, and the high *computational cost* to simulate operational model dynamics. Model error hinders the accuracy of forecasts, while the computational cost to simulate the dynamics hinders the quantification of uncertainties in these forecasts. Both challenges can be alleviated by leveraging data to learn a surrogate model for the dynamics. Data-driven methods enable learning closure terms and unresolved scales in the dynamics, thus enhancing the forecast skill of existing models. In addition, surrogate models are inexpensive to simulate and enable using a large number of particles within ensemble Kalman or Monte Carlo methods for state reconstruction and forecasting, thus enhancing the uncertainty quantification.

This paper investigates a framework for state reconstruction and forecasting that relies on data-driven surrogate modeling of the dynamics in a low-dimensional latent space. Our reduced-order autodifferentiable ensemble Kalman filters (ROAD-EnKFs) leverage the EnKF algorithm to estimate by maximum likelihood the latent dynamics as well as a decoder from latent space to state space. The learned latent dynamics and decoder are subsequently used to reconstruct and forecast the state. Numerical experiments show that, compared to existing methods, ROAD-EnKFs achieve higher accuracy at lower computational cost provided that the state dynamics exhibit a hidden low-dimensional structure. When such structure is not expressed in the latent dynamics, ROAD-EnKFs achieve similar accuracy at lower cost, making them a promising approach for surrogate state reconstruction and forecasting.

Our work blends in an original way several techniques and insights from inverse problems, data assimilation, machine learning, and reduced-order modeling. First, if the state dynamics were known and inexpensive to simulate, a variety of filtering and smoothing algorithms from data assimilation (e.g. extended, ensemble, and unscented Kalman filters and smoothers, as well as particle filters) can be used to reconstruct and forecast the state. These algorithms often build on a Bayesian formulation, where posterior inference on the state combines the observed data with a prior distribution defined using the model dynamics. Hence, learning a surrogate model for the dynamics can be interpreted as learning a prior regularization for state reconstruction and forecasting. Second, the task of learning the regularization can be viewed as an inverse problem: we seek to recover the state dynamics from partially and noisily observed trajectories. Data assimilation facilitates the numerical solution of this inverse problem by providing estimates of the hidden state. Third, our work leverages machine learning and reduced-order modeling to parameterize the dynamics in a low-dimensional latent space and learn a decoder from latent space to state space. In particular, we parameterize the decoder using recent ideas from discretization-invariant operator learning. Our numerical experiments demonstrate the computational advantage of co-learning an inexpensive surrogate model in latent space together with a decoder, rather than a more expensive-to-simulate dynamics in state space.

4.1.1 Related Work

Ensemble Kalman Filters in Data Assimilation The EnKF algorithm, reviewed in [Houtekamer and Zhang, 2016; Katzfuss et al., 2016; Roth et al., 2017; Sanz-Alonso et al., 2018], is a popular method for state reconstruction and forecasting in data assimilation, with applications in numerical weather forecasting, the geophysical sciences, and signal processing

[Evensen, 1994, 2009; Szunyogh et al., 2008; Whitaker et al., 2008]. The EnKF propagates N equally-weighted particles through the dynamics, and assimilates new observations via Kalman-type updates computed with empirical moments. If the state dynamics are known, the EnKF can achieve accurate reconstruction with a small ensemble size N even in applications where the state and the observations are high-dimensional, provided that the *effective dimension* is moderate [Ghattas and Sanz-Alonso, 2022]; EnKFs with a small ensemble size have a low computational and memory cost compared to traditional Kalman filters [Roth et al., 2017]. Ensemble Kalman methods are also successful solvers for inverse problems, as reviewed in [Chada et al., 2021]. In this paper, we employ the EnKF to approximate the data log-likelihood of surrogate models for unknown or expensive-to-simulate dynamics. The use of the EnKF for maximum likelihood estimation (MLE) was first proposed in [Stroud et al., 2010], which adopted a derivative-free optimization approach; see also [Pulido et al., 2018]. Empirical studies on the likelihood computed with EnKFs and other data assimilation techniques can be found in [Carrassi et al., 2017; Metref et al., 2019]. The application of the EnKF to approximate the data log-likelihood within pseudo-marginal Markov chain Monte Carlo methods for Bayesian parameter estimation was investigated in [Drovandi et al., 2021]; see also [Stroud and Bengtsson, 2007; Stroud et al., 2018]. The paper [Chen et al., 2022] introduced derivative-based optimization of an EnKF approximation of the log-likelihood to perform state and parameter estimation in high-dimensional nonlinear systems. However, to the best of our knowledge, no prior work combines estimation of the log-likelihood via EnKFs with learning low-dimensional surrogate models, including both surrogate latent dynamics and a decoder from latent space to state space.

Blending Data Assimilation with Reduced-Order Models Model reduction techniques have been employed in data assimilation to improve the state reconstruction accuracy in high-dimensional dynamical systems. The assimilation in the unstable subspace (AUS) method [Carrassi et al., 2008; Trevisan et al., 2010; Palatella et al., 2013; Sanz-Alonso and

Stuart, 2015; Law et al., 2016a] projects the dynamics onto a time-dependent subspace of the tangent space where the dynamics are unstable, and assimilates the observations therein. The unstable directions are determined by the Lyapunov vectors with nonnegative Lyapunov exponents, and can be approximated using discrete QR algorithms [Dieci and Van Vleck, 2007, 2015]. The observations can also be projected onto the unstable directions to reduce the data dimension [Maclean and Van Vleck, 2021]. We refer to [Benner et al., 2015] for a review of projection-based model reduction techniques. However, these methods rely on prior knowledge about the dynamics to identify the unstable subspaces and to construct the latent dynamics, and data assimilation is performed *after* the subspaces are found. In contrast, our paper introduces a framework that uses data assimilation as a tool to build surrogate latent dynamics from data. Another approach to reduce the dimension of data assimilation problems exploits the conditionally Gaussian distribution of slow variables arising in the stochastic parameterization of a wide range of dynamical systems [Chen and Qi, 2022; Chen and Majda, 2018; Majda and Qi, 2018]. This conditional Gaussian structure can be exploited to obtain adequate uncertainty quantification of forecasts with a moderate sample size. A caveat, however, is that identifying the slow variables can be challenging in practice. As in our approach, these techniques often rely on machine learning to learn closure terms for the dynamics [Chen and Qi, 2022]. Finally, we refer to [Spantini et al., 2018] for a discussion on how the effective dimension of transport map methods for data assimilation can be reduced by exploiting the conditional independence structure of the reference-target pair.

Merging Data Assimilation with Machine Learning Recent developments in machine learning to model dynamical systems from data are reviewed in [Levine and Stuart, 2021]. One line of work [Tandeo et al., 2015; Ueno and Nakamura, 2014; Dreano et al., 2017; Pulido et al., 2018] embeds the EnKF and the ensemble Kalman smoother (EnKS) into the expectation-maximization (EM) algorithm for MLE [Dempster et al., 1977], with a special focus on estimation of error covariance matrices. The expectation step (E-step) is approximated by EnKF/EnKS with the Monte Carlo EM objective [Wei and Tanner, 1990]. A subsequent line of work [Nguyen et al., 2019; Brajard et al., 2020; Bocquet et al., 2020; Farchi et al., 2021; Wikner et al., 2020] introduces training of a neural network (NN) surrogate model in the maximization step (M-step) based on the states filtered by the E-step. Unfortunately, it can be hard to achieve an accurate approximation of the E-step using EnKF/EnKS [Chen et al., 2022]. Another line of work [Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018; Corenflos et al., 2021] approximates the data log-likelihood with particle filters (PFs) [Gordon et al., 1993; Doucet and Johansen, 2009] and performs MLE using derivative-based optimization. However, the resampling step in PFs is not readily differentiable, and, in addition, PFs often collapse when the dimensions of the state and the observations are large [Agapiou et al., 2017; Bengtsson et al., 2008]. Finally, techniques that leverage machine learning to obtain inexpensive *analog* ensembles for data assimilation are starting to emerge [Yang and Grooms, 2021].

Data-Driven Modeling of Dynamical Systems with Machine Learning Machine learning is also useful for dimensionality reduction in time-series modeling. As an important example, recurrent neural networks (RNNs) [Lipton et al., 2015; Yu et al., 2019] assimilate data into the time-evolving latent states using NN updates. The paper [Rubanova et al., 2019] models the latent state evolution in recurrent networks with NN-embedded differential equations [Chen et al., 2018]. Other types of NN updates to incorporate the data into latent states include gated recurrent units (GRU) [De Brouwer et al., 2019; Jordan et al., 2021], long short-term memory (LSTM) [Lechner and Hasani, 2020; Harlim et al., 2021], and controlled differential equations (CDEs) [Kidger et al., 2020]. Another approach is to directly model the differential equation governing the state dynamics from observation data using regression. Such methods include sparse regression over a dictionary of candidate functions using L_1-regularization [Brunton et al., 2016; Tran and Ward, 2017; Schaeffer et al., 2018]. These techniques rely on full observation of the state, and, importantly, on time-derivative data that are rarely available in practice and are challenging to approximate from noisy discrete-time observation data [He et al., 2022b,a]. When the data are not guaranteed to lie in the same space as the underlying dynamics, an autoencoder structure can be jointly learned with the latent state dynamics [Champion et al., 2019]. Different modeling techniques can be applied to learn the latent state dynamics, including sparse dictionary regression [Champion et al., 2019], recurrent networks [Gonzalez and Balajewicz, 2018; Maulik et al., 2021], and the Koopman operator learning [Lusch et al., 2018]. It is important to notice that, in contrast to, e.g., [Brunton et al., 2016; He et al., 2022c], the focus of this paper is on state reconstruction and forecasting, rather than on obtaining an interpretable model for the dynamics.

4.1.2 Outline and Main Contributions

- §4.2 formalizes the problem setting and goals. We introduce a reduced-order state-space model (SSM) framework, where the dynamics are modeled in a low-dimensional latent space and learned jointly with a decoder from latent space to state space.

- §4.3 introduces our main algorithm, the reduced-order autodifferentiable ensemble Kalman filter (ROAD-EnKF). As part of the derivation of the algorithm, we discuss the use of EnKFs to estimate the data log-likelihood within reduced-order SSMs.

- §4.4 contains important implementation considerations, including the design of the decoder, the use of truncated backpropagation to enhance the scalability for large windows of data, and the choice of regularization in latent space.

- §4.5 demonstrates the performance of our method in three examples: (i) a Lorenz 63 model embedded in a high-dimensional space, where we compare our approach to the SINDy-AE algorithm [Champion et al., 2019]; (ii) Burgers equation, where we showcase that ROAD-EnKFs are able to forecast the emergence of shocks, a phenomenon not included in our training data-set; and (iii) Kuramoto-Sivashinky equation, a common test problem for filtering methods due to its chaotic behavior, where the ROAD-EnKF framework provides a computational benefit over state-of-the-art methods with similar accuracies.

- §4.6 closes with a summary of the paper and open questions for further research.

Notation

We denote by $t \in \{0, 1, \dots\}$ a discrete-time index and by $n \in \{1, \dots, N\}$ a particle index. Time indices will be denoted with subscripts and particles with superscripts, so that u_t^n represents a generic particle n at time t. We denote the particle dimension by d_u. We denote $u_{t_0:t_1} := \{u_t\}_{t=t_0}^{t_1}$ and $u^{n_1:n_2} := \{u^n\}_{n=n_0}^{n_1}$. The collection $u_{t_0:t_1}^{n_0:n_1}$ is defined similarly. The Gaussian density with mean m and covariance C evaluated at u is denoted by $\mathcal{N}(u; m, C)$. The corresponding Gaussian distribution is denoted by $\mathcal{N}(m, C)$.

4.2 Problem Formulation

In this section, we formalize and motivate our goals: reconstructing and forecasting a time-evolving, partially-observed state with unknown or expensive-to-simulate dynamics. An important step towards these goals is to learn a surrogate model for the dynamics. In Subsection 4.2.1 we consider an SSM framework where the state dynamics are parameterized and learned in order to reconstruct and forecast the state. Next, in Subsection 4.2.2, we introduce a reduced-order SSM framework where the dynamics are modeled in a latent space and a decoder from latent space to state space is learned along with the latent dynamics. Our ROAD-EnKF algorithm, introduced in Section 4.3, operates in this reduced-order SSM.

4.2.1 Setting and Motivation

Consider a parameterized SSM of the form:

$$
\begin{aligned}
&\text{(dynamics)} && u_t = F_\alpha(u_{t-1}) + \xi_t, && \xi_t \sim \mathcal{N}(0, Q_\beta), && 1 \le t \le T, && (4.2.1)\\
&\text{(observation)} && y_t = H_t u_t + \eta_t, && \eta_t \sim \mathcal{N}(0, R_t), && 1 \le t \le T, && (4.2.2)\\
&\text{(initialization)} && u_0 \sim p_u(u_0). && && && (4.2.3)
\end{aligned}
$$

The state dynamics map $F_\alpha : \mathbb{R}^{d_u} \to \mathbb{R}^{d_u}$ and error covariance matrix $Q_\beta \in \mathbb{R}^{d_u \times d_u}$ depend on unknown parameter $\theta := (\alpha^\top, \beta^\top)^\top \in \mathbb{R}^{d_\theta}$. The observation matrices $H_t \in \mathbb{R}^{d_y \times d_u}$ and error covariance matrices $R_t \in \mathbb{R}^{d_y \times d_y}$ are assumed to be known and possibly time-varying. We further assume independence of all random variables u_0, $\xi_{1:T}$, and $\eta_{1:T}$.

Given observation data $y_{1:T}$ drawn from the SSM (4.2.1)-(4.2.3), we aim to accomplish two goals:

Goal 1: Reconstruct the states $u_{1:T}$.

Goal 2: Forecast the states $u_{T+1:T+T_f}$ for some forecast lead time $T_f \ge 1$.

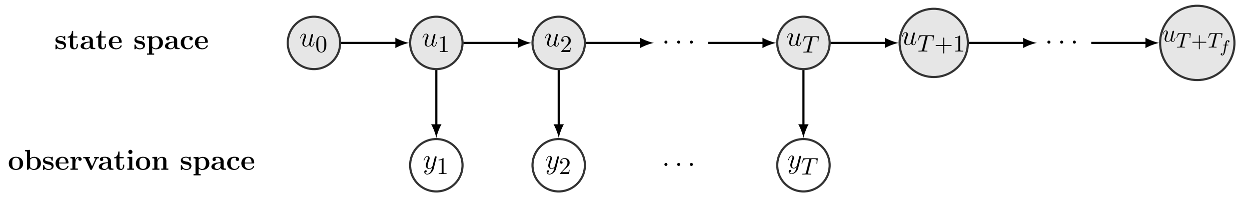


Figure 4.1: Structure of data under SSM (4.2.1)-(4.2.3), where we assume only observations $y_{1:T} := \{y_1, \ldots, y_T\}$ are available. Our goals are to reconstruct the states $u_{1:T}$ (Goal 1) and to forecast future states $u_{T+1:T+T_f}$ for some $T_f \ge 1$ (Goal 2).

If the true parameter $\theta \in \mathbb{R}^{d_\theta}$ was known and the dynamics were inexpensive to simulate, the first goal can be accomplished by applying a filtering (or smoothing) algorithm on the SSM (4.2.1)-(4.2.3), while the second goal can be accomplished by iteratively applying the dynamics model Eq. (4.2.1) to the reconstructed state u_T. We are interested in the case where θ needs to be estimated in order to reconstruct and forecast the state.

The covariance Q_β in the dynamics model (4.2.1) may represent model error or stochastic forcing in the dynamics; in either case, estimating Q_β from data can improve the reconstruction and forecast of the state. In this paper, we are motivated by applications where F_α represents a surrogate model for the flow between observations of an autonomous ordinary differential equation (ODE). Letting Δ_s be the equally-spaced time between observations and $f_\alpha : \mathbb{R}^{d_u} \mapsto \mathbb{R}^{d_u}$ be the parameterized vector field of the differential equation, we then have

$$\text{(ODE)} \qquad \frac{\mathrm{d}u}{\mathrm{d}s} = f_\alpha(u), \qquad F_\alpha : u(s) \mapsto u(s + \Delta_s), \tag{4.2.4}$$

where $u(s) \in \mathbb{R}^{d_u}$ is the state as a function of continuous-time variable $s \geq 0$. The ODE (4.2.4) may arise from spatial discretization of a system of partial differential equations (PDEs). For instance, we will consider 1-dimensional partial differential equations for $u(x, s)$ of order $\kappa \geq 1$, where u is a function of the spatial variable $x \in [0, \mathsf{L}]$ and continuous-time variable $s \geq 0$:

$$\text{(PDE)} \qquad \frac{\partial u}{\partial s} = f_\alpha\biggl(u, \frac{\partial u}{\partial x}, \ldots, \frac{\partial^\kappa u}{\partial x^\kappa}\biggr), \qquad F_\alpha : u(\cdot, s) \mapsto u(\cdot, s + \Delta_s), \tag{4.2.5}$$

with suitable boundary conditions. After discretizing this equation on a spatial domain with grid points $0 = x_1 < x_2 < \cdots < x_M = \mathsf{L}$, Eq. (4.2.5) can be expressed in the form of Eq. (4.2.4) by replacing the spatial derivatives with their finite difference approximations, and u, f_α, F_α with their finite-dimensional approximations on the grid. As a result, d_u equals the number of grid points M. Several examples and additional details will be given in §4.5.

4.2.2 Reduced-Order Modeling

When the state is high-dimensional (i.e., d_u is large), direct reconstruction and forecast of the state is computationally expensive, and surrogate modeling of the state dynamics map F_α becomes challenging. We then advocate reconstructing and forecasting the state

u_t through a low-dimensional latent representation z_t, modeling the state dynamics within the low-dimensional latent space. This idea is formalized via the following reduced-order parameterized SSM:

$$
\begin{aligned}
&\text{(latent dynamics)} && z_t = G_\alpha(z_{t-1}) + \zeta_t, && \zeta_t \sim \mathcal{N}(0, S_\beta), && 1 \le t \le T, && (4.2.6)\\
&\text{(decoding)} && u_t = D_\gamma(z_t), && && 1 \le t \le T, && (4.2.7)\\
&\text{(observation)} && y_t = H_t u_t + \eta_t, && \eta_t \sim \mathcal{N}(0, R_t), && 1 \le t \le T, && (4.2.8)\\
&\text{(latent initialization)} && z_0 \sim p_z(z_0). && && && (4.2.9)
\end{aligned}
$$

The latent dynamics map $G_\alpha : \mathbb{R}^{d_z} \mapsto \mathbb{R}^{d_z}$ and error covariance matrix $S_\beta \in \mathbb{R}^{d_z \times d_z}$ are defined on a d_z dimensional latent space with $d_z < d_u$, and the decoder function $D_\gamma : \mathbb{R}^{d_z} \mapsto \mathbb{R}^{d_u}$ maps from latent space to state space. The reduced-order SSM depends on an unknown parameter $\theta := (\alpha^\top, \beta^\top, \gamma^\top)^\top \in \mathbb{R}^{d_\theta}$. The remaining assumptions are the same as in §4.2.1.

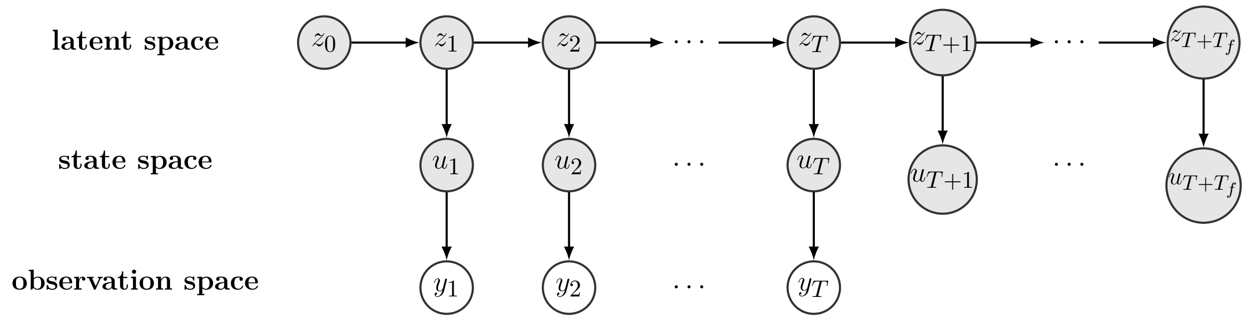


Figure 4.2: Structure of data under reduced-order SSM (4.2.6)-(4.2.9), where we assume only observations $y_{1:T} := \{y_1, \dots, y_T\}$ are available. Our goals are to reconstruct the states $u_{1:T}$ (Goal 1) and to forecast future states $u_{T+1:T+T_f}$ for some $T_f \ge 1$ (Goal 2).

Writing $\mathcal{H}_{\gamma,t}(\cdot) := H_t D_\gamma(\cdot)$, the reduced-order SSM (4.2.6)-(4.2.9) can be combined into

$$
\begin{aligned}
&\text{(latent dynamics)} && z_t = G_\alpha(z_{t-1}) + \zeta_t, && \zeta_t \sim \mathcal{N}(0, S_\beta), && 1 \le t \le T, && (4.2.10)\\
&\text{(observation)} && y_t = \mathcal{H}_{\gamma,t}(z_t) + \eta_t, && \eta_t \sim \mathcal{N}(0, R_t), && 1 \le t \le T, && (4.2.11)\\
&\text{(latent initialization)} && z_0 \sim p(z_0), && && && (4.2.12)
\end{aligned}
$$

where the observation function $\mathcal{H}_{\gamma,t}(\cdot)$ is nonlinear if the decoder $D_\gamma(\cdot)$ is nonlinear. As in Subsection 4.2.1, the map G_α may be interpreted as the flow between observations of an ODE with vector field $g_\alpha : \mathbb{R}^{d_z} \mapsto \mathbb{R}^{d_z}$.

If the true parameter $\theta \in \mathbb{R}^{d_\theta}$ was known, given observation data $y_{1:T}$ drawn from the reduced-order SSM (4.2.6)-(4.2.9), we can reconstruct the states $u_{1:T}$ (Goal 1) by first applying a filtering (or smoothing) algorithm on (4.2.10)-(4.2.12) to estimate $z_{1:T}$, and then applying the decoder D_γ. We can forecast the states $u_{T+1:T+T_f}$ (Goal 2) by first applying iteratively the latent dynamics model Eq. (4.2.6) to the reconstructed latent state z_T, and then applying the decoder D_γ. As in §4.2.1, we are interested in the case where θ needs to be estimated from the given data $y_{1:T}$.

4.3 Reduced-Order Autodifferentiable Ensemble Kalman Filters

As discussed in the previous section, to achieve both goals of state reconstruction and forecast, it is essential to obtain a suitable surrogate model for the dynamics by learning the parameter θ. The general approach we take is the following: (1) estimate θ with maximum likelihood; (2) apply a filtering algorithm with estimated parameter θ to reconstruct and forecast the states $u_{1:T}$. As we shall see, the maximum likelihood estimation of θ will rely itself on a filtering algorithm. For the SSM in §4.2.1, this approach was introduced in [Chen et al., 2022] via AD-EnKF (Algorithm 4.1 in [Chen et al., 2022]). Here we focus on the reduced-order SSM in §4.2.2, namely (4.2.10)-(4.2.12), which is a more general case than the SSM in §4.2.1; this explains the terminology reduced-order AD-EnKF (ROAD-EnKF).

In §4.3.1, we describe how the log-likelihood $\mathcal{L}(\theta) = \log p_\theta(y_{1:T})$ can be expressed in terms of the normalizing constants that arise from sequential filtering. In §4.3.2, we give background on EnKFs and on how to use these filtering algorithms to estimate $\mathcal{L}(\theta)$. In §4.3.3, we introduce our ROAD-EnKF method that takes as input multiple independent instances of observation data $y^{\mathsf{I}}_{1:T}$ across the same time range, and performs both state reconstruction and forecasting.

4.3.1 Sequential Filtering and Data Log-Likelihood

Suppose that $\theta = (\alpha^\top, \beta^\top, \gamma^\top)^\top$ is known. We recall that, for $1 \le t \le T$, the *filtering distributions* $p_\theta(z_t|y_{1:t})$ of the SSM (4.2.10)-(4.2.12) can be obtained sequentially, alternating between *prediction* and *analysis* steps:

$$\text{(prediction)} \qquad p_\theta(z_t|y_{1:t-1}) = \int \mathcal{N}\big(z_t; G_\alpha(z_{t-1}), S_\beta\big) p_\theta(z_{t-1}|y_{1:t-1})\, \mathrm{d}z_{t-1}, \tag{4.3.1}$$

$$\text{(analysis)} \qquad p_\theta(z_t|y_{1:t}) = \frac{1}{\mathcal{E}_t(\theta)} \mathcal{N}\big(y_t; \mathcal{H}_{\gamma,t}(z_t), R_t\big) p_\theta(z_t|y_{1:t-1}), \tag{4.3.2}$$

with the convention $p_\theta(\cdot|y_{1:0}) := p_\theta(\cdot)$. Here $\mathcal{E}_t(\theta)$ is a normalizing constant which does not depend on z_t. It can be shown that

$$\mathcal{E}_t(\theta) = p_\theta(y_t|y_{1:t-1}) = \int \mathcal{N}\big(y_t; \mathcal{H}_{\gamma,t}(z_t), R_t\big) p_\theta(z_t|y_{1:t-1})\, \mathrm{d}z_t, \tag{4.3.3}$$

and therefore the data log-likelihood admits the characterization

$$\mathcal{L}(\theta) := \log p_\theta(y_{1:T}) = \sum_{t=1}^{T} \log p_\theta(y_t|y_{1:t-1}) = \sum_{t=1}^{T} \log \mathcal{E}_t(\theta). \tag{4.3.4}$$

Analytical expressions of the filtering distributions $p_\theta(z_t|y_{1:t})$ and the data log-likelihood $\mathcal{L}(\theta)$ are only available for a small class of SSMs, which includes linear-Gaussian and discrete SSMs [Kalman, 1960; Papaspiliopoulos et al., 2014]. Outside these special cases, filtering algorithms need to be employed to approximate the filtering distributions, and these algorithms can be leveraged to estimate the log-likelihood.

4.3.2 Estimation of the Log-Likelihood with Ensemble Kalman Filters

Given $\theta = (\alpha^\top, \beta^\top, \gamma^\top)^\top$, the EnKF algorithm [Evensen, 1994, 2009] sequentially approximates the filtering distributions $p_\theta(z_t|y_{1:t})$ using N equally-weighted particles $z_t^{1:N}$. At prediction steps, each particle z_t^n is propagated using the latent dynamics model Eq. (4.2.10), while at analysis steps a Kalman-type update is performed for each particle:

$$\text{(prediction step)} \qquad \widehat{z}_t^n = G_\alpha(z_{t-1}^n) + \zeta_t^n, \qquad \zeta_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, S_\beta), \tag{4.3.5}$$

$$\text{(analysis step)} \qquad z_t^n = \widehat{z}_t^n + \widehat{K}_t\big(y_t + \eta_t^n - \mathcal{H}_{\gamma,t}(\widehat{z}_t^n)\big), \qquad \eta_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, R_t). \tag{4.3.6}$$

The Kalman gain $\widehat{K}_t := \widehat{C}_{zy,t}(\widehat{C}_{yy,t} + R_t)^{-1}$ is defined using empirical covariances given by

$$\widehat{C}_{zy,t} = \frac{1}{N-1}\sum_{n=1}^N (\widehat{z}_t^n - \widehat{m}_t)\big(\mathcal{H}_{\gamma,t}(\widehat{z}_t^n) - \widehat{\mathcal{H}}_t\big)^\top, \quad \widehat{C}_{yy,t} = \frac{1}{N-1}\sum_{n=1}^N \big(\mathcal{H}_{\gamma,t}(\widehat{z}_t^n) - \widehat{\mathcal{H}}_t\big)\big(\mathcal{H}_{\gamma,t}(\widehat{z}_t^n) - \widehat{\mathcal{H}}_t\big)^\top, \tag{4.3.7}$$

where

$$\widehat{m}_t = \frac{1}{N}\sum_{n=1}^N \widehat{z}_t^n, \qquad \widehat{\mathcal{H}}_t = \frac{1}{N}\sum_{n=1}^N \mathcal{H}_{\gamma,t}(\widehat{z}_t^n). \tag{4.3.8}$$

The empirical moments $\widehat{C}_{yy,t}, \widehat{\mathcal{H}}_t$ defined in equations Eq. (4.3.7) and Eq. (4.3.8) provide a Gaussian approximation to the *predictive distribution* for $\mathcal{H}_{\gamma,t}(z_t)$:

$$p_\theta(\mathcal{H}_{\gamma,t}(z_t)|y_{1:t-1}) \approx \mathcal{N}(\mathcal{H}_{\gamma,t}(z_t); \widehat{\mathcal{H}}_t, \widehat{C}_{yy,t}). \tag{4.3.9}$$

By applying the change of variables formula to Eq. (4.3.3), we have

$$\begin{aligned} \mathcal{E}_t(\theta) &= \int \mathcal{N}\big(y_t; \mathcal{H}_{\gamma,t}(z_t), R_t\big) p_\theta(z_t|y_{1:t-1})\,\mathrm{d}z_t \\ &= \int \mathcal{N}\big(y_t; \mathcal{H}_{\gamma,t}(z_t), R_t\big) p_\theta(\mathcal{H}_{\gamma,t}(z_t)|y_{1:t-1})\,\mathrm{d}\mathcal{H}_{\gamma,t}(z_t) \\ &\approx \mathcal{N}(y_t; \widehat{\mathcal{H}}_t, \widehat{C}_{yy,t} + R_t), \end{aligned} \tag{4.3.10}$$

where the approximation step follows from Eq. (4.3.9) and the formula for convolution of two Gaussians. From Eq. (4.3.4), we obtain the following estimate of the data log-likelihood:

$$\mathcal{L}_{\text{EnKF}}(\theta) := \sum_{t=1}^{T} \log \mathcal{N}\big(y_t; \widehat{\mathcal{H}}_t, \widehat{C}_{yy,t} + R_t\big) \approx \mathcal{L}(\theta). \tag{4.3.11}$$

The estimate $\mathcal{L}_{\text{EnKF}}(\theta)$ can be computed online with EnKF, and is stochastic as it depends on the randomness used to propagate the particles, e.g., the choice of random seed. The whole procedure is summarized in Algorithm 4.3.1, which implicitly defines a stochastic map $\theta \mapsto \mathcal{L}_{\text{EnKF}}(\theta)$.

Algorithm 4.3.1 Ensemble Kalman Filter and Log-likelihood Estimation

```
   Input: $\theta = (\alpha^\top, \beta^\top, \gamma^\top)^\top, y_{1:T}$. (If multiple input instances $y^{\mathsf{I}}_{1:T}$ are provided, run the following procedure for each instance $y^i_{1:T}$.)
1: Initialize $\mathcal{L}_{\text{EnKF}}(\theta) = 0$. Draw $z_0^n \overset{\text{i.i.d.}}{\sim} p_z(z_0)$.
2: for $t = 1, \dots, T$ do
3:     Set $\widehat{z}_t^n = G_\alpha(z_{t-1}^n) + \zeta_t^n$, where $\zeta_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, S_\beta)$.        ▷ Prediction step
4:     Compute $\widehat{m}_t, \widehat{\mathcal{H}}_t, \widehat{C}_{zy,t}, \widehat{C}_{yy,t}$ by equations Eq. (4.3.7) and Eq. (4.3.8) and set $\widehat{K}_t = \widehat{C}_{zy,t}(\widehat{C}_{yy,t} + R_t)^{-1}$.
5:     Set $z_t^n = \widehat{z}_t^n + \widehat{K}_t\big(y_t + \eta_t^n - \mathcal{H}_{\gamma,t}(\widehat{z}_t^n)\big)$, where $\eta_t^n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, R_t)$.        ▷ Analysis step
6:     Set $\mathcal{L}_{\text{EnKF}}(\theta) \leftarrow \mathcal{L}_{\text{EnKF}}(\theta) + \log \mathcal{N}\big(y_t; \widehat{\mathcal{H}}_t, \widehat{C}_{yy,t} + R_t\big)$.
7: end for
   Output: EnKF particles $z_{0:T}^{1:N}$. Log-likelihood estimate $\mathcal{L}_{\text{EnKF}}(\theta)$.(If multiple input instances $y^{\mathsf{I}}_{1:T}$ are provided, return instead the average of log-likelihood estimates.)
```

4.3.3 Main Algorithm

The main idea of our algorithm is to perform maximum likelihood estimation on the parameter θ by gradient ascent, via differentiation through the map $\theta \mapsto \mathcal{L}_{\text{EnKF}}(\theta)$. Our core method is summarized in Alg. 4.3.2, which includes estimation of θ as well as reconstruction and forecast of states. Our PyTorch implementation is at `https://github.com/ymchen0/ROAD-EnKF`. The gradient of the map $\theta^k \mapsto \mathcal{L}_{\text{EnKF}}(\theta^k)$ can be evaluated using autodiff libraries [Paszke et al., 2019; Bradbury et al., 2018; Abadi et al., 2016] that support auto-differentiation of common matrix operations, e.g. matrix multiplication, inverse, and determinant [Giles, 2008a]. We use the “reparameterization trick” [Kingma and Welling, 2014; Rezende et al., 2014] to auto-differentiate through the stochasticity in the EnKF algorithm, as in Subsection 4.1 of [Chen et al., 2022].

In Section 4.5, we consider numerical examples where the data are generated from an unknown SSM in the form of Eq. (4.2.6)-Eq. (4.2.9) with no explicit knowledge of the reduced-order structure; we also consider examples where the data are generated directly from Eq. (4.2.1)-Eq. (4.2.3). In practice, multiple independent instances of observation data $y_{1:T}^{\mathsf{I}}$ may be available across the same time range, where each superscript $i \in \mathsf{I}$ corresponds to one instance of observation data $y_{1:T}$. We assume that each instance $y_{1:T}^{i}$ is drawn i.i.d. from the same SSM, with different realizations of initial state, model error, and observation error for each instance. We assume that data are split into training and test sets $y_{1:T}^{\mathsf{I}_{\text{train}}}$ and $y_{1:T}^{\mathsf{I}_{\text{test}}}$. During training, we randomly select a small batch of data from $y_{1:T}^{\mathsf{I}_{\text{train}}}$ at each iteration, and evaluate the averaged log-likelihood and its gradient over the batch to perform a parameter update. The idea is reminiscent of stochastic gradient descent in the optimization literature: matrix operations of EnKF can be parallelized within a batch to utilize the data more efficiently, reducing the computational and memory cost compared to using the full training set at each iteration. The state reconstruction and forecast performance are evaluated on the unseen test set $y_{1:T}^{\mathsf{I}_{\text{test}}}$.

State reconstruction and forecast via Alg. 4.3.2 can be interpreted from a probabilistic point of view. For convenience, we drop the superscripts I and k in this discussion. For $0 \leq t \leq T$, since the particles $z_t^{1:N}$ form an approximation of the filtering distribution $p_\theta(z_t|y_{1:t})$ for latent state z_t, it follows from Eq. (4.2.7) that the output particles $u_t^{1:N}$ of the algorithm form an approximation of the filtering distribution $p_\theta(u_t|y_{1:t})$ for state u_t. For $T+1 \leq t \leq T+T_f$, it follows from Eq. (4.2.10) that the particles $z_t^{1:N}$ form an approximation of the predictive distribution $p_\theta(z_t|y_{1:T})$. Therefore, by Eq. (4.2.7) the output particles $u_t^{1:N}$ of the algorithm form an approximation of the predictive distribution $p_\theta(u_t|y_{1:T})$ for future state u_t.

Algorithm 4.3.2 Reduced-Order Autodifferentiable Ensemble Kalman Filter (ROAD-EnKF)

```
    Input: Observations $y^{\mathsf{I}}_{1:T}$, split into $y^{\mathsf{I}_{\text{train}}}_{1:T}$ and $y^{\mathsf{I}_{\text{test}}}_{1:T}$. Learning rate $\eta$. Batch size $B$.
 1: Initialize SSM parameter $\theta^0$ and set $k = 0$. Write $\mathcal{H}_{\gamma,t}(\cdot) = H_t D_\gamma(\cdot)$.
    // Training phase
 2: while not converging do
 3:     Randomly select $B$ indices from $\mathsf{I}_{\text{train}}$, denoted as $\mathsf{I}_B$.
 4:     Compute $z^{\mathsf{I}_B,1:N}_{0:T}, \mathcal{L}_{\text{EnKF}}(\theta^k) = \textsc{EnsembleKalmanFilter}(\theta^k, y^{\mathsf{I}_B}_{1:T})$ using algorithm Alg. 4.3.1.
 5:     Compute $\nabla_\theta \mathcal{L}_{\text{EnKF}}(\theta^k)$ by auto-differentiating the map $\theta^k \mapsto \mathcal{L}_{\text{EnKF}}(\theta^k)$.
 6:     Set $\theta^{k+1} = \theta^k + \eta \nabla_\theta \mathcal{L}_{\text{EnKF}}(\theta^k)$ and $k \leftarrow k + 1$.
 7: end while
    // Test phase
 8: $z^{\mathsf{I}_{\text{test}},1:N}_{0:T}, \mathcal{L}_{\text{EnKF}}(\theta^k) = \textsc{EnsembleKalmanFilter}(\theta^k, y^{\mathsf{I}_{\text{test}}}_{1:T})$.   ▷ State reconstruction
 9: Simulate $z^{\mathsf{I}_{\text{test}},1:N}_t$ using Eq. (4.2.10) with $\alpha = \alpha_k, \beta = \beta_k$ for $t = T+1, \dots, T+T_f$.   ▷ State forecast
10: Compute $u^{\mathsf{I}_{\text{test}},1:N}_{0:T+T_f} = D_{\gamma_k}(z^{\mathsf{I}_{\text{test}},1:N}_{0:T+T_f})$ .
    Output: Learned reduced-order SSM parameter $\theta^k$ and particles $u^{\mathsf{I}_{\text{test}},1:N}_{0:T+T_f}$.
```

4.4 Implementation Details

This section considers the practical implementation of ROAD-EnKF Alg. 4.3.2, including parameterization of the surrogate latent dynamics map g_α and decoder D_γ (§4.4.1), computational efficiency for high-dimensional observations (§4.4.2), and regularization on latent states (§4.4.3).

4.4.1 Surrogate Latent Dynamics and Decoder Design

In our numerical experiments, we adopt a simple parameterization for the surrogate latent dynamics map g_α using a two-layer fully connected NN. For our design of the decoder D_γ, the idea stems from the literature on convolutional autoencoders for computer vision tasks (e.g., [Mao et al., 2016]), where both the encoder and decoder networks consist of multiple convolutional layers with residual connections that map between the image space and latent space. Here, to suit our setting, we replace the kernel-based local convolutional layers with Fourier-based spectral convolutional layers ('Fourier layers') introduced in [Li et al., 2020b; Guibas et al., 2021]. The latter treat a finite-dimensional vector as a spatial discretization of a function on a grid, and learn a finite-dimensional mapping that approximates an operator between function spaces. The learning accuracy is known empirically to not depend on the level of the discretization [Li et al., 2020b], determined by d_u in our case. Using Fourier layers to learn dynamical systems and differential equations was originally proposed in [Li et al., 2020b]. For the sake of completeness, we describe below the definition of spectral convolutional layers and how they are incorporated into our decoder design.

Spectral Convolutional Layer Given an input $v_{\text{in}} \in \mathbb{R}^{n_{\text{in}} \times d_u}$ where n_{in} is the number of input channels and d_u is the input dimension, which is also the size of the grid where the function is discretized, we first apply a discrete Fourier transform (DFT) in spatial domain to get $\lambda_{\text{in}} := \text{DFT}(v_{\text{in}}) \in \mathbb{C}^{n_{\text{in}} \times d_u}$. We then multiply it by a learned complex weight tensor $W \in \mathbb{C}^{n_{\text{out}} \times n_{\text{in}} \times d_u}$ that is even symmetric[1] to get $\lambda_{\text{out}} := W \times \lambda_{\text{in}} \in \mathbb{C}^{n_{\text{out}} \times d_u}$. The multiplication is defined by

$$(W \times \lambda_{\text{in}})_{i,k} = \sum_{j=1}^{n_{\text{in}}} W_{i,j,k} (\lambda_{\text{in}})_{j,k}. \tag{4.4.1}$$

1. That is, W satisfies $W_{i,j,k} = \overline{W}_{i,j,d_u+2-k}$ $\forall i, j$ and $\forall k \geq 2$. This ensures that the inverse discrete Fourier transform of λ_{out} is real. In practice, the parameterization of W requires up to $n_{\text{in}} \times n_{\text{out}} \times (\lfloor d_u/2 \rfloor + 1)$ complex entries.

This can be regarded as 'channel mixing', since for the k-th Fourier mode ($1 \leq k \leq d_u$), all n_{in} input channels of λ are linearly mixed to produce n_{out} output channels through the matrix $W_{\cdot,\cdot,k}$. Other types of (possibly nonlinear) mixing introduced in [Guibas et al., 2021] can also be applied, and we leave them to future work. We then apply an inverse discrete Fourier transform (IDFT) in spatial domain to get the output $v_{\text{out}} = \text{IDFT}(\lambda_{\text{out}}) \in \mathbb{R}^{n_{\text{out}} \times d_u}$. We call the mapping $v_{\text{in}} \mapsto v_{\text{out}}$ a spectral convolutional layer (SpecConv).

Fourier Neural Decoder Given latent variable $z \in \mathbb{R}^{d_z}$ (where we omit the subscript t for convenience), we first apply a complex linear layer $f_0(\cdot)$ to get $z_0 = f_0(z) := W_0 z + b_0 \in \mathbb{C}^h$ for $W_0 \in \mathbb{C}^{h \times d_z}$ and $b_0 \in \mathbb{C}^h$, where h is the dimension of z_0 to be specified. We then apply an IDFT that treats z_0 as a one-sided Hermitian signal in Fourier domain[2] to get $v_0 := \text{IDFT}(z_0) \in \mathbb{R}^{d_u}$. We then apply L spectral convolutional layers to get v_L, with proper choices of channel numbers as well as residual connections, normalization layers, and activation functions. More specifically, v_L is defined by iteratively applying the following

$$v_\ell = f_\ell(v_{\ell-1}) := \text{Act}\Big(\text{Norm}\Big(\text{SpecConv}(v_{\ell-1}) + \text{1x1Conv}(v_{\ell-1})\Big)\Big), \qquad 1 \leq \ell \leq L, \tag{4.4.2}$$

where $v_\ell \in \mathbb{R}^{n_\ell \times d_u}$, Act and Norm refer to the activation function and the normalization layer, 1x1Conv refers to the one-by-one convolutional layer which can be viewed as a generalization of residual connection, and $n_0 = 1$. We refer to f_ℓ as a 'Fourier layer'. The final part of the decoder is a two-layer fully connected NN that is applied to $v_L \in \mathbb{R}^{n_L \times d_u}$ over channel dimension to get $u \in \mathbb{R}^{d_u}$. See Fig. 4.3 for the architecture. Notice that the learned variables γ of the decoder include W_0, b_0 of the initial linear layer, complex weight tensors W's of SpecConv layers, weights and biases of 1x1Conv layers, as well as the final fully-connected NN.

2. z_0 is either truncated or zero-padded to a signal of dimension $\mathbb{C}^{\lfloor d_u/2 \rfloor + 1}$.

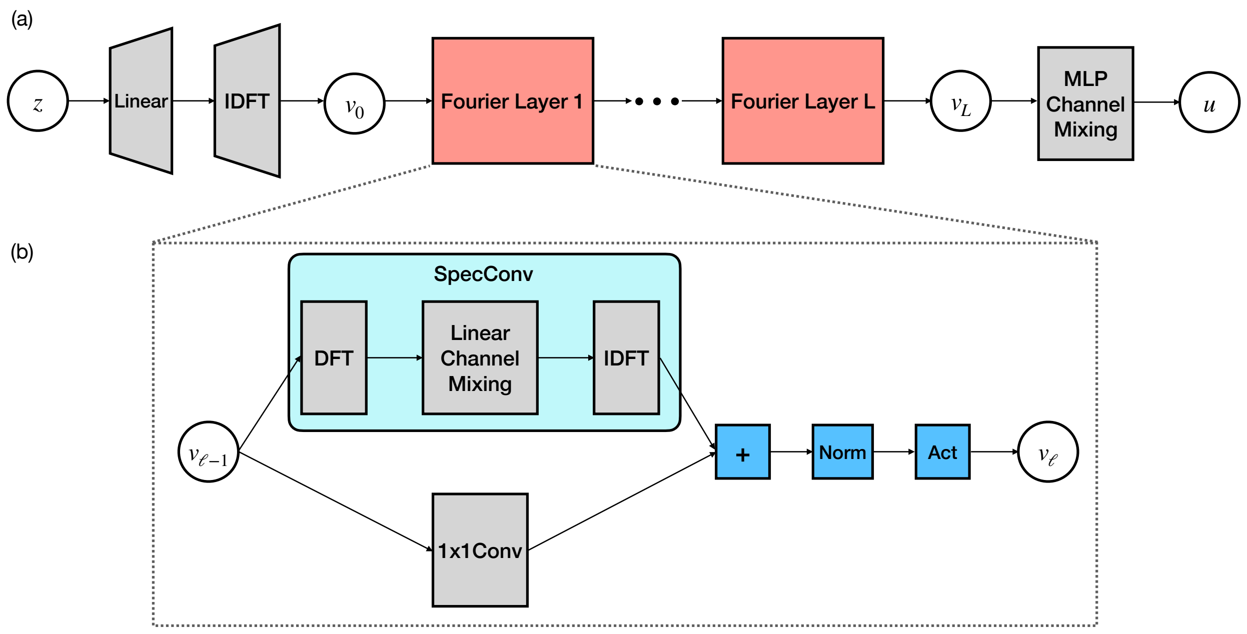


Figure 4.3: (a) **Network architecture of the decoder D_γ.** Starting from $z \in \mathbb{R}^{d_z}$ in a low-dimensional latent space, we first apply a complex linear layer followed by an IDFT to lift it to $v_0 \in \mathbb{R}^{d_u}$ in a high-dimensional state space. We then apply L Fourier layers iteratively to get $v_L \in \mathbb{R}^{n_L \times d_u}$ where n_L is the channel dimension. We project it back to the state space by applying a two-layer fully-connected NN to mix the channels and output $u \in \mathbb{R}^{d_u}$. (b) **Fourier layer**: The design was first proposed in [Li et al., 2020b], and we describe it here for the sake of completeness. The upper half represents a spectral convolutional layer, where we transform the input $v_{\ell-1} \in \mathbb{R}^{n_{\ell-1} \times d_u}$ into the frequency space with DFT, mix the channels with a complex linear map, and transform back with IDFT. The lower half is a one-by-one convolutional layer, which is a generalization of residual connection. The outputs from both layers are summed up and passed through a normalization and an activation layer to produce the output $v_\ell \in \mathbb{R}^{n_\ell \times d_u}$.

4.4.2 Algorithmic Design for Computational Efficiency

If the time-window length T is large, we follow [Chen et al., 2022] and use truncated back-propagation to auto-differentiate the map $\theta \mapsto \mathcal{L}_{\text{EnKF}}(\theta)$: we divide the sequence into multiple short subsequences and backpropagate within each subsequence. The idea stems from Truncated Backpropagation Through Time (TBPTT) for RNNs [Williams and Zipser, 1995; Sutskever et al., 2014] and the recursive maximum likelihood method for hidden Markov models [Le Gland and Mevel, 1997]. By doing so, multiple gradient ascent steps can be performed for each single filtering pass, and thus the data can be utilized more efficiently.

Moreover, gradient explosion/vanishing [Bengio et al., 1994] are less likely to happen. We refer to [Chen et al., 2022] for more details. We choose this variant of ROAD-EnKF in our experiments.

In this work we are mostly interested in the case where d_u and d_y are large, and d_z is small. Moreover, the ensemble size N that we consider is moderate, i.e., $d_u \geq d_y > N > d_z$. Therefore, we do not pursue the covariance localization approach as in [Chen et al., 2022] (see also [Houtekamer and Mitchell, 2001; Hamill et al., 2001]), which is most effective when $N < d_z$. Instead, we notice that the computational bottlenecks of the analysis step in the EnKF Alg. 4.3.1 are the $O(d_y^3)$ operations of computing the Kalman gain (Line 4) as well as updating the data log-likelihood (Line 6), where we need to compute the matrix inverse and log-determinant of a $d_y \times d_y$ matrix $(\widehat{C}_{yy,t} + R_t)$. If $d_y > N$, the number of operations can be improved to $O(N^3)$ as follows. Let $Y_t \in \mathbb{R}^{d_y \times N}$ be the matrix representation of the centered ensemble after applying the observation function, i.e., its n-th column is $Y_t^n := \frac{1}{\sqrt{N-1}}\big(\mathcal{H}_{\gamma,t}(\widehat{z}_t^n) - \frac{1}{N}\sum_{m=1}^N \mathcal{H}_{\gamma,t}(\widehat{z}_t^m)\big)$ (we drop the parameter γ for convenience). This leads to $\widehat{C}_{yy,t} = Y_t Y_t^\top$. By the matrix inversion lemma [Woodbury, 1950],

$$(\widehat{C}_{yy,t} + R_t)^{-1} = R_t^{-1} - R_t^{-1} Y_t (I + Y_t^\top R_t^{-1} Y_t)^{-1} Y_t^\top R_t^{-1}, \tag{4.4.3}$$

$$\operatorname{logdet}(\widehat{C}_{yy,t} + R_t) = \operatorname{logdet}(I + Y_t^\top R_t^{-1} Y_t) + \operatorname{logdet}(R_t), \tag{4.4.4}$$

where $I + Y_t^\top R_t^{-1} Y_t \in \mathbb{R}^{N \times N}$. The computational cost can be further reduced if the quantities R_t^{-1} and $\operatorname{logdet}(R_t)$ can be pre-computed, for instance when $R_t = rI$ for some scalar $r \in \mathbb{R}$.

Moreover, in practice, to update the ensemble in Line 5 of Alg. 4.3.1, instead of inverting $I + Y_t^\top R_t^{-1} Y_t$ directly in Eq. (4.4.3) followed by a matrix multiplication, we find it more numerically stable to first solve the following linear system:

$$(I + Y_t^\top R_t^{-1} Y_t) u_t^n = Y_t^\top R_t^{-1}\big(y_t + \gamma_t^n - \mathcal{H}_{\gamma,t}(\widehat{z}_t^n)\big) \tag{4.4.5}$$

for $u_t^n \in \mathbb{R}^N$, and then perform the analysis step (Line 5 of Alg. 4.3.1) by

$$\begin{aligned} z_t^n &= \widehat{z}_t^n + \widehat{C}_{zy,t}(\widehat{C}_{yy,t} + R_t)^{-1}\big(y_t + \gamma_t^n - \mathcal{H}_{\gamma,t}(\widehat{z}_t^n)\big) \\ &= \widehat{z}_t^n + \widehat{C}_{zy,t}\big(R_t^{-1} - R_t^{-1}Y_t(I + Y_t^\top R_t^{-1}Y_t)^{-1}Y_t^\top R_t^{-1}\big)\big(y_t + \gamma_t^n - \mathcal{H}_{\gamma,t}(\widehat{z}_t^n)\big) \\ &= \widehat{z}_t^n + \widehat{C}_{zy,t}R_t^{-1}\big(y_t + \gamma_t^n - \mathcal{H}_{\gamma,t}(\widehat{z}_t^n) - Y_t u_t^n\big). \end{aligned} \tag{4.4.6}$$

Similar ideas and computational cost analysis can be found in [Tippett et al., 2003]. For the benchmark experiments in §4.5, we modify the AD-EnKF algorithm as presented in [Chen et al., 2022] to incorporate the above ideas.

4.4.3 Latent Space Regularization

Since the estimation of u is given by $D_\gamma(z)$, where both $D_\gamma(\cdot)$ and z need to be identified from data, we overcome potential identifiability issues by regularizing z in the latent space. To further motivate the need for latent space regularization, consider the following example: if the pair $(z, D_\gamma(\cdot))$ provides a good estimation of u_t, then so does $(cz, \frac{1}{c}D_\gamma(\cdot))$ for any constant $c \neq 0$. Therefore, the norm of z can be arbitrarily large, and thus we regularize z's in the latent space so that their norms do not explode.

We perform regularization by extending the observation model Eq. (4.2.11) to impose additional constraints on the latent state variable z_t's. The idea stems from regularization in ensemble Kalman methods for inverse problems [Chada et al., 2020; Guth et al., 2022]. We first extend Eq. (4.2.11) to the equations:

$$\begin{cases} y_t = \mathcal{H}_{\gamma,t}(z_t) + \eta_t, & \eta_t \sim \mathcal{N}(0, R_t), \\ 0 = z_t + \epsilon_t, & \epsilon_t \sim \mathcal{N}(0, \sigma^2 I_{d_z}), \end{cases} \tag{4.4.7}$$

where σ is a parameter to be chosen that incorporates the prior information that each coordinate of z_t is an independent centered Gaussian random variable with standard deviation

σ. Define

$$y_t^{\text{aug}} = \begin{bmatrix} y_t \\ 0 \end{bmatrix}, \qquad \mathcal{H}_{\gamma,t}^{\text{aug}}(z_t) = \begin{bmatrix} \mathcal{H}_{\gamma,t}(z_t) \\ z_t \end{bmatrix}, \qquad \eta_t^{\text{aug}} \sim \mathcal{N}(0, R_t^{\text{aug}}), \qquad R_t^{\text{aug}} = \begin{bmatrix} R_t & 0 \\ 0 & \sigma^2 I_{d_z} \end{bmatrix}. \tag{4.4.8}$$

We then write Eq. (4.4.7) into an augmented observation model

$$y_t^{\text{aug}} = \mathcal{H}_{\gamma,t}^{\text{aug}}(z_t) + \eta_t^{\text{aug}}, \qquad \eta_t^{\text{aug}} \sim \mathcal{N}(0, R_t^{\text{aug}}). \tag{4.4.9}$$

To perform latent space regularization in ROAD-EnKF, during the training stage we run EnKF (Line 4 of Alg. 4.3.2) with augmented data $y_{1:T}^{\text{aug}}$ and SSM with the augmented observation model, i.e., (4.2.10)-(4.4.9)-(4.2.12). During test stage, we run EnKF (Line 8 of Alg. 4.3.2) with the original data and SSM, i.e., (4.2.10)-(4.2.11)-(4.2.12).

4.5 Numerical Experiments

In this section, we compare our ROAD-EnKF method to the SINDy autoencoder [Champion et al., 2019], which we abbreviate as SINDy-AE. It learns an encoder-decoder pair that maps between observation space (y_t's) and latent space (z_t's), and simultaneously performs a sparse dictionary learning in the latent space to discover the latent dynamics. Similar to SINDy-AE, our ROAD-EnKF method jointly discovers a latent space and the dynamics therein that is a low-dimensional representation of the data. However, our method differs from SINDy-AE in four main aspects: (1) No time-derivative data for $y_{1:T}$ are required; (2) No encoder is required; (3) State reconstruction and forecast can be performed even when the data $y_{1:T}$ are noisy and partial observation of $u_{1:T}$, while SINDy-AE is targeted at noiseless and fully observed data that are dense in time; (4) Stochastic representation of latent dynamics model can be learned, and uncertainty quantification can be performed in state reconstruction and forecast tasks through the use of particles, while SINDy-AE only provides a point estimate in both tasks.

We also compare our ROAD-EnKF method to AD-EnKF [Chen et al., 2022]. Although AD-EnKF enjoys some of the benefits of ROAD-EnKF, including the capability to learn from noisy, partially observed data and perform uncertainty quantification, it directly learns the dynamics model in high-dimensional state space (i.e., on u_t's instead of z_t's), which leads to higher model complexity, as well as additional computational and memory costs when performing the EnKF step. Moreover, AD-EnKF does not take advantage of the possible low-dimensional representation of the state. We compare in Table 4.5.1 below the capabilities of the three algorithms under different scenarios.

	Learn from noisy and partially observed data	Uncertainty quantification	No need of time-derivative data	Low-dimensional state representation
SINDy-AE [Champion et al., 2019]	✗	✗	✗	✓
AD-EnKF [Chen et al., 2022]	✓	✓	✓	✗
ROAD-EnKF (this paper)	✓	✓	✓	✓

Table 4.5.1: Comparison of SINDy-AE, AD-EnKF, and ROAD-EnKF under different scenarios.

Other alternative methods include EnKF-embedded EM algorithms (e.g. [Brajard et al., 2020]) and autodifferentiable PF algorithms (e.g., [Naesseth et al., 2018]). Since [Chen et al., 2022] already establishes AD-EnKF's superiority to those approaches, we do not include them in these experiments, and we refer to [Chen et al., 2022] for more details.

The training procedure is the following: We first specify a forecast lead time T_f. We then generate training data $y_{0:T}^{\mathsf{I}_{\text{train}}}$ and test data with extended time range $(u_{0:T+T_f}^{\mathsf{I}_{\text{test}},*}, y_{0:T}^{\mathsf{I}_{\text{test}}})$ with $N_{\text{train}} := |\mathsf{I}_{\text{train}}|$ and $N_{\text{test}} := |\mathsf{I}_{\text{test}}|$. The data are either generated from a reduced-order SSM Eq. (4.2.6)-Eq. (4.2.9) with explicit knowledge of true parameter θ (§4.5.1), or from an SSM Eq. (4.2.1)-Eq. (4.2.3) with no explicit knowledge of the exact reduced-order structure (Subsections 4.5.2 and 4.5.3). The data $y_{0:T}^{\mathsf{I}_{\text{train}}}$ and $y_{0:T}^{\mathsf{I}_{\text{test}}}$ are then passed into ROAD-EnKF (Alg. 4.3.2), and we evaluate the following:

Reconstruction-RMSE (RMSE-r): Measures the state reconstruction error of the algorithm. We take the particle mean of $u_{0:T}^{\mathsf{I}_{\text{test}},1:N}$ as a point estimate of the true states $u_{0:T}^{\mathsf{I}_{\text{test}},*}$, and evaluate the RMSE:

$$\text{RMSE-r} = \sqrt{\frac{1}{d_u N_{\text{test}}(T - T_b)} \sum_{t=T_b}^{T} \sum_{i \in \mathsf{I}_{\text{test}}} \left| \overline{u}_t^i - u_t^{i,*} \right|^2}, \qquad \text{where } \overline{u}_t^i = \frac{1}{N} \sum_{n=1}^{N} u_t^{i,n}. \quad (4.5.1)$$

Here T_b is a number of burn-in steps to remove transient errors in the reconstruction that stem from the choice of initialization. For simplicity, we set $T_b = \lfloor T/5 \rfloor$ as in [Chen et al., 2022].

Forecast-RMSE (RMSE-f): Measures the t-step state forecast error of the algorithm, for lead time $t \in \{1, \dots, T_f\}$. We take the particle mean of $u_{T+t}^{\mathsf{I}_{\text{test}},1:N}$ as a point estimate of the true future states $u_{T+t}^{\mathsf{I}_{\text{test}},*}$:

$$\text{RMSE-f}(t) = \sqrt{\frac{1}{d_u N_{\text{test}}} \sum_{i \in \mathsf{I}_{\text{test}}} \left| \overline{u}_{T+t}^i - u_{T+t}^{i,*} \right|^2}, \qquad \text{where } \overline{u}_{T+t}^i = \frac{1}{N} \sum_{n=1}^{N} u_{T+t}^{i,n}. \quad (4.5.2)$$

Test Log-Likelihood: Measures the averaged log-likelihood of the learned reduced-order SSM over test observation data $y_{0:T}^{\mathsf{I}_{\text{test}}}$, which is $\mathcal{L}_{\text{EnKF}}(\theta^k)$ defined in Line 8 of Alg. 4.3.2.

For AD-EnKF, the above metrics can be similarly computed, following [Chen et al., 2022]. For SINDy-AE, as uncertainty quantification is not performed, we use its decoder output as the point estimate of the state in both reconstruction and forecast. Moreover, log-likelihood computation is not available for SINDy-AE.

4.5.1 Embedding of Chaotic Dynamics (Lorenz 63)

In this subsection, we reconstruct and forecast a state defined by embedding a Lorenz 63 (L63) model in a high-dimensional state space. A similar experiment was used in [Champion et al., 2019] to motivate the SINDy-AE algorithm, and hence this example provides a good point of comparison. The data are generated using the L63 system as the true latent state dynamics model:

$$\frac{\mathrm{d}z}{\mathrm{d}s} = g(z), \qquad \begin{cases} g^{(1)}(z) = 10(z^{(2)} - z^{(1)}), \\ g^{(2)}(z) = z^{(1)}(28 - z^{(3)}) - z^{(2)}, \\ g^{(3)}(z) = z^{(1)}z^{(2)} - \frac{8}{3}z^{(3)}, \end{cases} \qquad G : z(s) \mapsto z(s + \Delta_s), \tag{4.5.3}$$

where $z^{(i)}$ and $g^{(i)}$ denote the i-th coordinate of z and component of g, and Δ_s is the time between observations. We further assume there is no noise in the true latent state dynamics model, i.e., $S = 0$. To construct the true reduced-order SSM, we define $D \in \mathbb{R}^{d_u \times 6}$ such that its i-th column $D^i \in \mathbb{R}^{d_u}$ is given by the discretized i-th Legendre polynomial over d_u grid points. The true states $u_t \in \mathbb{R}^{d_u}$ are defined by

$$u_t := D \begin{bmatrix} z_t^{(1)}/40 & z_t^{(2)}/40 & z_t^{(3)}/40 & (z_t^{(1)}/40)^3 & (z_t^{(2)}/40)^3 & (z_t^{(3)}/40)^3 \end{bmatrix}^\top. \tag{4.5.4}$$

We consider two cases of the observation model (4.2.8): (1) full observation, where all coordinates of u_t are observed, i.e., $H_t = I_{d_u}$ and $d_y = d_u$; (2) partial observation, where for each t, only a fixed portion $c < 1$ of all coordinates of u_t are observed, and the coordinate indices are chosen randomly without replacement. In this case, $H_t \in \mathbb{R}^{d_y \times d_u}$ is a submatrix of I_{d_u} and varies across time, and $d_y = cd_u$. This partial observation set-up has been studied in the literature (e.g., [Brajard et al., 2020; Bocquet et al., 2020]) for data assimilation problems. For both cases, we assume $R_t = 0.01 I_{d_y}$ and $z_0 \sim \mathcal{N}(0, 4I_{d_z})$.

We consider full observation with $d_u = d_y = 128$ and partial observation with $d_u = 128$, $d_y = 64$ (i.e., $c = 1/2$). We generate $N_{\text{train}} = 1024$ training data and $N_{\text{test}} = 20$ test data with the true reduced-order SSM defined by (4.5.3) and (4.5.4). We set the number of observations $T = 250$ with time between observations $\Delta_s = 0.1$. We set the forecast lead time $T_f = 10$. The latent flow map G is integrated using the Runge–Kutta–Fehlberg method. The surrogate latent dynamics map g_α is parameterized as a two-layer fully connected NN, and is integrated using a fourth-order Runge-Kutta method with step size $\Delta_s^{\text{int}} = 0.05$. The error covariance matrix S_β in the latent dynamics is parametrized using a diagonal matrix with positive diagonal elements $\beta \in \mathbb{R}^{d_z}$. The decoder D_γ is parameterized as a Fourier Neural Decoder (FND) discussed in §4.4.1. Details of the network hyperparameters for this and subsequent examples are summarized in Table 4.5.2, obtained through cross-validation experiments on the training dataset. The latent space dimension for both SINDy-AE and ROAD-EnKF is set to $d_z = 3$. The ensemble size for both AD-EnKF and ROAD-EnKF is set to $N = 100$.

		L63	Burgers	KS
FND	L	4	2	4
	h	6	40	40
	$(n_0, \ldots, n_L)$	(1, 20, 20, 20, 20)	(1, 20, 20)	(1, 20, 20, 20, 20)
	Norm	LayerNorm		
	Activation	ReLU		
Latent space reg.	σ	2	4	4
Optimization	Optimizer	Adam		
	Learning rate (η)	1e-3		
	Batch size (B)	16	4	4
	TBPTT length	10		

Table 4.5.2: Choices of hyperparameters for ROAD-EnKF on different numerical examples.

In Table 4.5.3 we list the performance metrics of each method with full and partial observation. The state reconstruction and forecast performance on a single instance of test data are plotted in Fig. 4.4 and 4.5 for the full observation case, and in Fig. 4.6 for the partial observation case. For the full observation case, we compare ROAD-EnKF with AD-EnKF and SINDy-AE, adopting for the latter the implementation in [Champion et al., 2019]. Since SINDy-AE requires time-derivative data as input, we use a finite difference approximation computed from data $y_{1:T}$. We also include the results for SINDy-AE where the exact time-derivative data are used. We find that ROAD-EnKF is able to reconstruct and forecast the states consistently with the lowest RMSE, and the performance is not affected by whether the state is fully or partially observed. AD-EnKF is able to reconstruct and forecast the state with a higher RMSE than that of ROAD-EnKF, and the performance deteriorates in the partially observed setting. SINDy-AE with finite difference approximation of derivative data also achieves higher reconstruction RMSE than that of ROAD-EnKF, and does not give accurate state forecasts. This is likely due to the fact that data are sparse in time (i.e., Δ_s is large) which leads to a larger error when approximating the true time-derivative, and hence it is more difficult to extract meaningful dynamics from the data. Even when the true time-derivative data are used (which is not available unless we have explicit knowledge of the true reduced-order SSM), SINDy-AE has a higher reconstruction RMSE compared to ROAD-EnKF, and its forecast performance is still worse than the other two methods. Moreover, it cannot handle partial observation.

In terms of computational cost, ROAD-EnKF is more efficient than AD-EnKF since the surrogate dynamics are cheaper to simulate and the EnKF algorithm is more efficient to perform in both training and testing. However, ROAD-EnKF takes more time than SINDy-AE, since the latter does not rely on a filtering algorithm, but rather an encoder, to reconstruct the states and perform learning.

	SINDy-AE (full)	SINDy-AE (w/ derivative, full)	AD-EnKF (full)	ROAD-EnKF (full)	AD-EnKF (partial)	ROAD-EnKF (partial)
RMSE-r	0.0142	0.0148	0.0168	0.0078	0.0368	0.0079
RMSE-f(1)	0.1310	0.0191	0.0156	0.0069	0.0315	0.0069
RMSE-f(5)	1.6580	0.0333	0.0335	0.0141	0.0729	0.0125
Log-likelihood	–		2.25×10^4	2.58×10^4	1.28×10^4	1.40×10^4
Training time (per epoch)	5.15s		9.74s	6.15s	8.86s	5.62s
Test time	2.35s		4.57s	2.95s	4.52s	2.73s

Table 4.5.3: Performance metrics for different algorithms at convergence. (Embedded L63 example, §4.5.1.)

4.5.2 Burgers Equation

In this subsection and the following one, we learn high-dimensional SSMs without explicit reference to a true model for low-dimensional latent dynamics. We first consider the 1-dimensional Burgers equation for $u(x, s)$, where u is a function of the spatial variable $x \in [0, \mathsf{L}]$ and continuous-time variable $s > 0$:

$$\begin{aligned} \frac{\partial u}{\partial s} &= -u\frac{\partial u}{\partial x} + \nu \frac{\partial^2 u}{\partial x^2}, \\ u(0, s) &= u(\mathsf{L}, s) = 0, \\ u(x, 0) &= u_0(x). \end{aligned} \tag{4.5.5}$$

Here ν is the viscosity parameter, and we set $\nu = 1/150$, $\mathsf{L} = 2$. Burgers equation [Burgers, 1948] has various applications in fluid dynamics, including modeling of viscous flows. We are interested in reconstructing solution states, as well as in the challenging problem of forecasting shocks that emerge outside the time range covered by the training data. Equation Eq. (4.5.5) is discretized on $[0, \mathsf{L}]$ with equally-spaced grid points $0 = x_1 < x_2 < \cdots < x_M = \mathsf{L}$, using a second-order finite difference method. Setting $\Delta x := x_i - x_{i-1} = \frac{\mathsf{L}}{M-1}$, we obtain

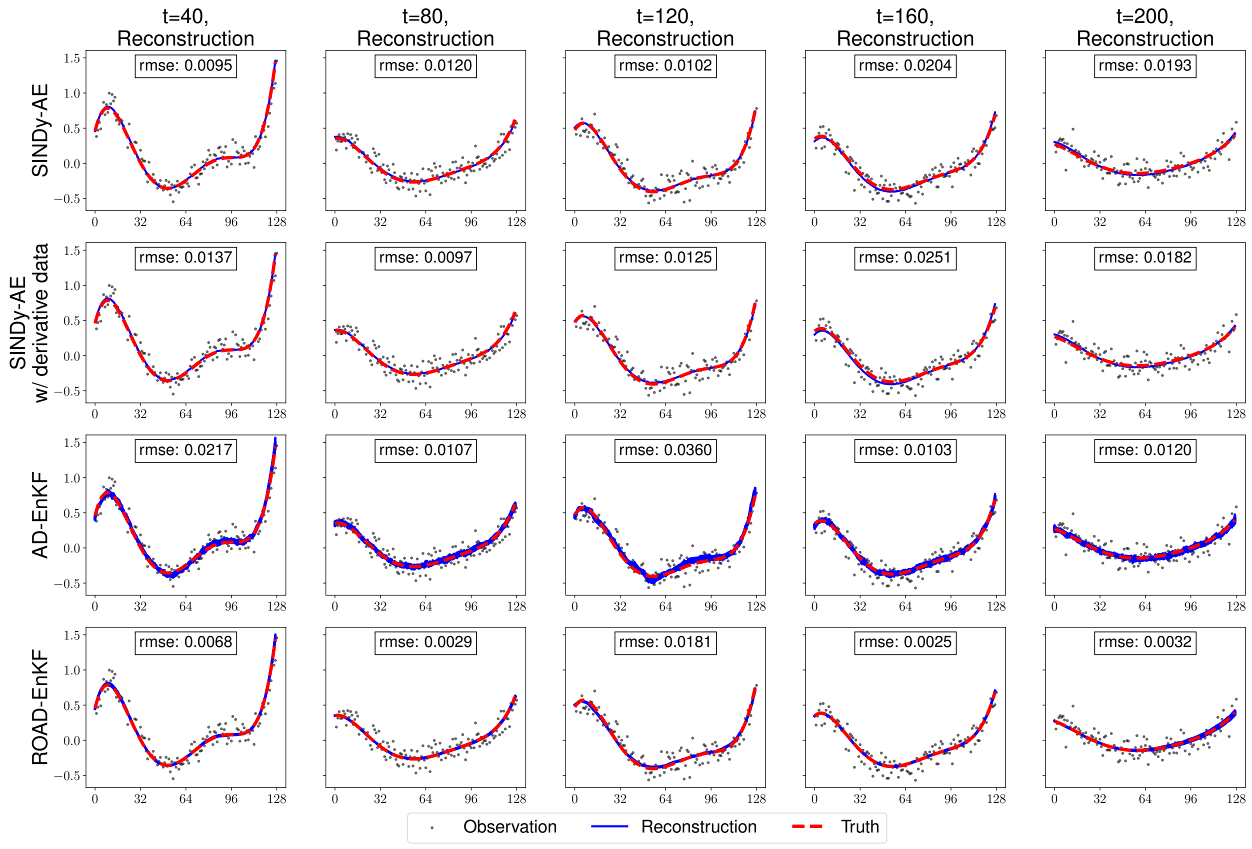


Figure 4.4: State reconstruction performance with full observation ($d_u = d_y = 128$) on the embedded L63 example in §4.5.1. For each method (row), the reconstructed states u_t (blue) for a single test sequence are plotted for $t = 40, 80, 120, 160, 200$ (column). The true values of the 128-dimensional states are plotted in red dashed lines, along with the noisy observations in black dots. Both AD-EnKF and ROAD-EnKF perform probabilistic state reconstructions through particles (all plotted in blue), while SINDy-AE only provides point estimates. The reconstruction RMSE's are computed for each plot. For SINDy-AE, even *with* derivative data (not required for AD-EnKF and ROAD-EnKF), the reconstruction performance is similar to that of AD-EnKF, while being worse than that of ROAD-EnKF.

the following ODE system:

$$
\begin{aligned}
&\frac{\mathrm{d}u^{(i)}}{\mathrm{d}s} = -\frac{\left(u^{(i+1)}\right)^2 - \left(u^{(i-1)}\right)^2}{4\Delta x} + \nu\frac{u^{(i+1)} - 2u^{(i)} + u^{(i-1)}}{\Delta x^2}, \qquad i = 2, \dots, M-1, \\
&u^{(1)}(s) = u^{(M)}(s) = 0, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (4.5.6) \\
&u^{(i)}(0) = u_0(i\Delta x).
\end{aligned}
$$

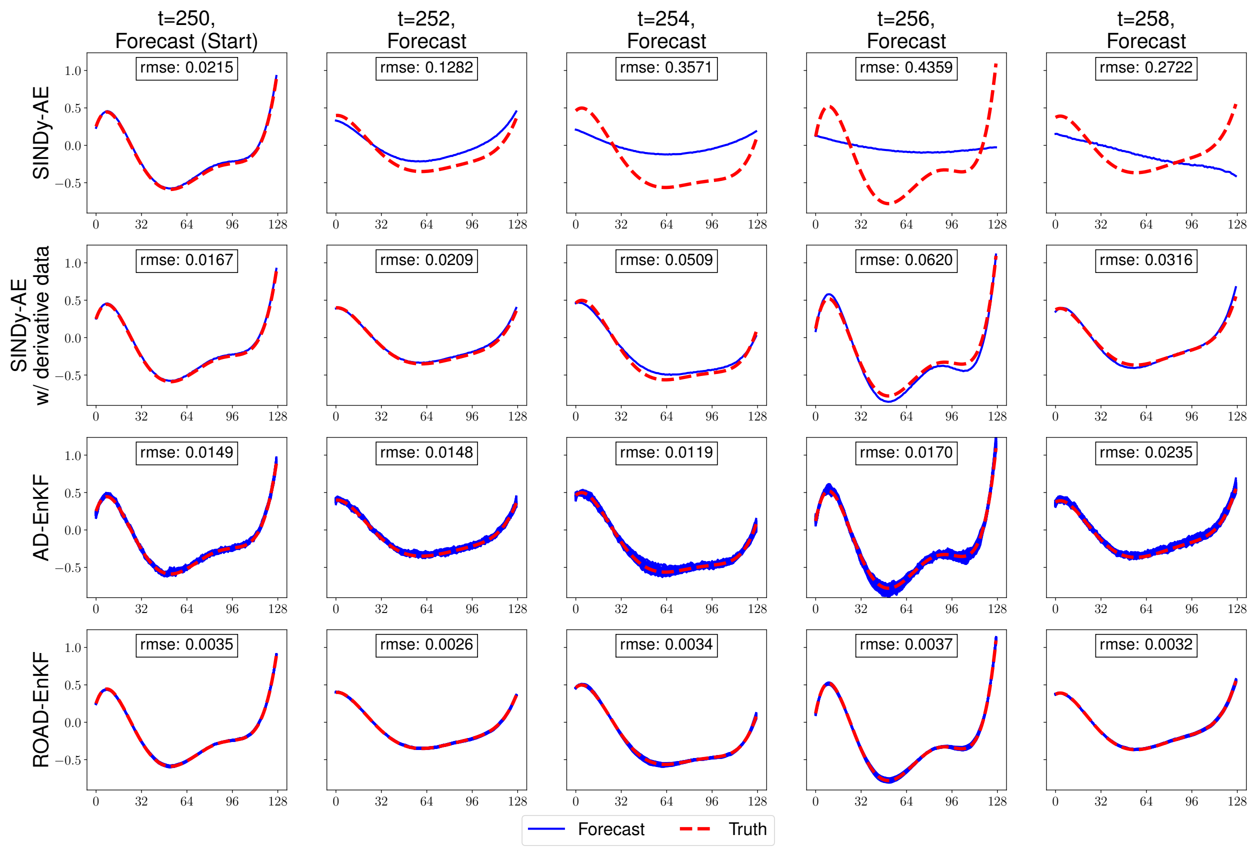


Figure 4.5: Forecast performance with full observation ($d_u = d_y = 128$) on the embedded L63 example in §4.5.1. For each method (row), the forecasted states u_t (blue) for a single test sequence are plotted for $t = 250$ (start of forecast), $252, 254, 256, 258$ (column). The true values of the $d_u = 128$ dimensional states are plotted in red dashed lines. Both AD-EnKF and ROAD-EnKF perform probabilistic forecast through particles (all plotted in blue), while SINDy-AE only provides point estimates. The forecast RMSE's are computed for each plot. For SINDy-AE, even *with* derivative data (not required for AD-EnKF and ROAD-EnKF), the forecast performance is similar to that of AD-EnKF, while being worse than that of ROAD-EnKF.

Here $u^{(i)}(s)$ is an approximation of $u(i\Delta x, s)$, the value of u at the i-th spatial node at time s. Equation Equation (4.5.6) defines a flow map $F : u(s) \mapsto u(s + \Delta_s)$ for state variable u with $d_u = M$, which we refer to as the true state dynamics model. We assume there is no noise in the dynamics, i.e., $Q = 0$.

Similar to §4.5.1, we consider two cases: full observation with $d_u = d_y = 256$ and partial observation with $d_u = 256$, $d_y = 128$ (i.e., $c = 1/2$). The initial conditions u_0 are generated

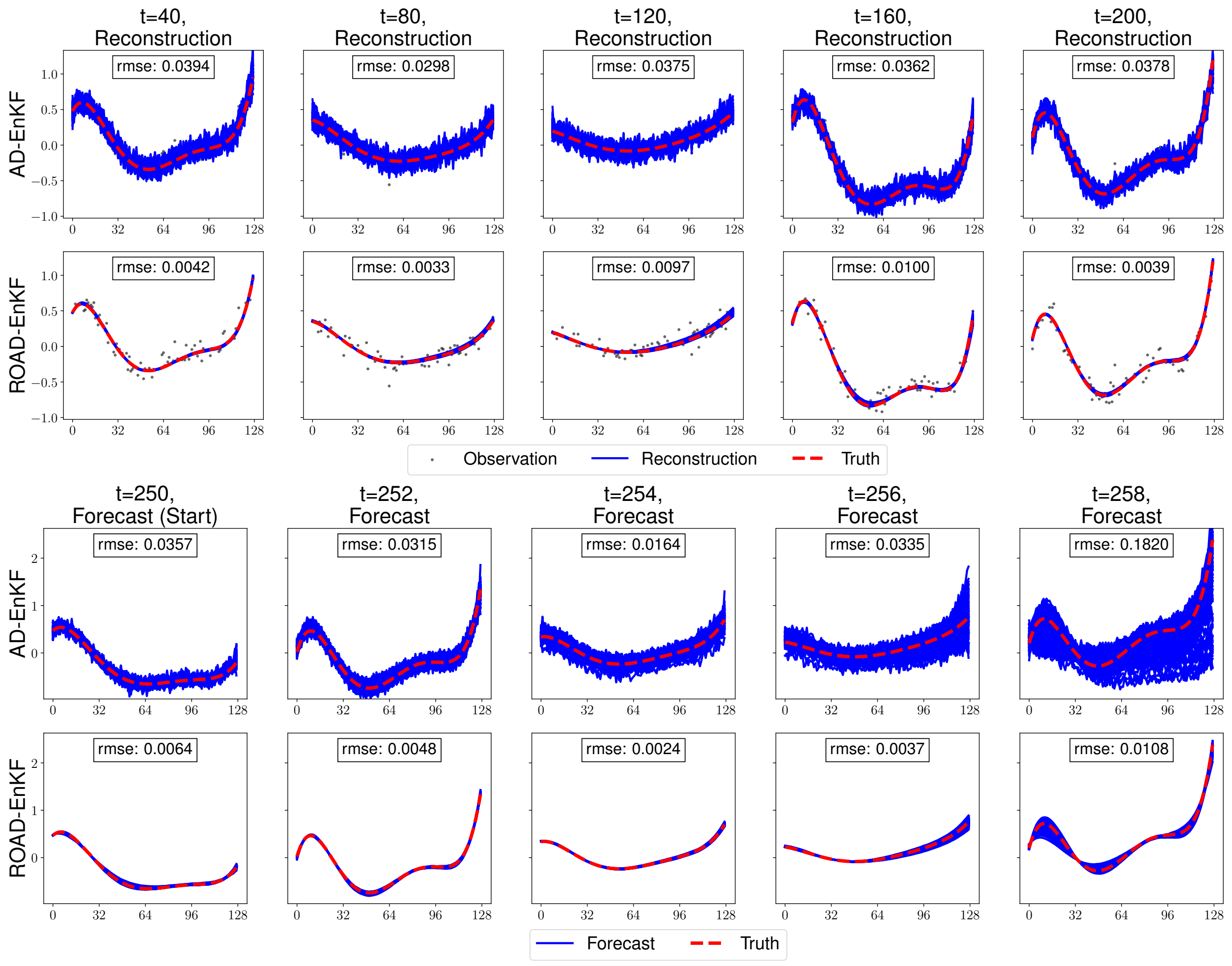


Figure 4.6: State reconstruction (upper half) and forecast (lower half) performance with partial observation ($d_u = 128$, $d_y = 64$) on the embedded L63 example in §4.5.1. For each method, the reconstructed states u_t (blue) for a single test sequence are plotted for $t = 40, 80, 120, 160, 200$ (column), and the forecasted states u_t (blue) for a single test sequence are plotted for $t = 250$ (start of forecast), $252, 254, 256, 258$ (column). The true values of the 128-dimensional states are plotted in red dashed lines, along with the noisy observations in black dots. SINDy-AE is inapplicable here because it cannot handle partial observations, while both AD-EnKF and ROAD-EnKF perform probabilistic state reconstructions and forecast through particles (all plotted in blue). The reconstruction/forecast RMSEs are computed for each plot.

in the following way:

$$u_0^{(i)} = U \sin \frac{2\pi i \Delta x}{\mathsf{L}}, \qquad U \sim \text{Uniform}(0.5, 1.5). \tag{4.5.7}$$

We generate $N_{\text{train}} = 1024$ training data and $N_{\text{test}} = 20$ test data with the true state dynamics model defined through Eqs. (4.5.6) and (4.5.7) with $R_t = 0.01 I_{d_y}$. We set the number of observations $T = 300$ with time between observations $\Delta_s = 0.001$. We set the forecast lead time $T_f = 300$. The flow map F is integrated using the fourth-order Runge–Kutta method with a fine step size $\Delta_s/20$. The surrogate latent dynamics map g_α is parameterized as a two-layer fully connected NN, and is integrated using a fourth-order Runge-Kutta method with step size $\Delta_s^{\text{int}} = 0.001$. The error covariance matrix S_β in the latent dynamics is parametrized using a diagonal matrix with positive diagonal elements $\beta \in \mathbb{R}^{d_z}$. The decoder D_γ is parameterized as an FND, discussed in §4.4.1. Details of the network hyperparameters are listed in Table 4.5.2. The latent space dimension for ROAD-EnKF is set to $d_z = 40$. The ensemble size for both AD-EnKF and ROAD-EnKF is set to $N = 100$. In this example and the following one, we set $z_0 \sim \mathcal{N}(0, \sigma^2 \mathsf{I}_{d_z})$ with the same σ defined in §4.4.3.

In Table 4.5.4, we list the performance metrics of each method with full and partial observation. The state reconstruction and forecast performance on a single instance of test data are plotted in Figures 4.7 (snapshots) and 4.8 (contour plot) for the partial observation case. Corresponding plots with full observation are shown in Figures 4.11 and 4.12 in the appendix. We find that ROAD-EnKF is able to reconstruct and forecast the states with the lowest RMSE, in both full and partial observation scenarios. More importantly, the emergence of shocks is accurately forecasted even though this phenomenon is not included in the time range covered by the training data. AD-EnKF achieves a higher RMSE than ROAD-EnKF for both state reconstruction and forecast tasks. AD-EnKF forecasts the emergence of shocks with lower accuracy than ROAD-EnKF, which indicates that AD-EnKF fails to fully learn the state dynamics. SINDy-AE with finite difference approximation of derivative data has the highest reconstruction RMSE among the three methods, and is not able to produce meaningful long-time state forecasts. This is remarkable, given that in this example the data are relatively dense (Δ_s is small) which facilitates, in principle, the approximation of time derivatives. In terms of computational cost, ROAD-EnKF is more efficient than AD-EnKF during both training and testing, but takes more time than SINDy-AE for the same reason as in §4.5.1.

In Table 4.5.5, we list the performance metrics of ROAD-EnKF with full observation and different choices of latent space dimension d_z ranging from 1 to 240. The results for partial observation show a similar trend and are not shown. We find that, as d_z increases, the state reconstruction performance stabilizes when $d_z \geq 4$. In order to achieve better long-time state forecast performance, d_z needs to be further increased, and the forecast performance stabilizes when $d_z \geq 10$. Both training and testing time slightly increase as d_z grows, which can be explained by the following: The computational time for both training and testing can be divided into the prediction step and the analysis step. We have shown in §4.4.2 that the computational bottleneck of the analysis step depends on the choices of ensemble size N and d_y, and is less affected by the increase of d_z. Moreover, the computational time of the prediction step depends on the complexity of the surrogate latent dynamics (two-layer NNs), which are relatively cheap to simulate for ROAD-EnKF. On the other hand, AD-EnKF enjoys similar computational complexity as ROAD-EnKF during the analysis step, but requires a more complicated surrogate model (NNs with Fourier layers) to capture the dynamics, which is more expensive to simulate. More experimental results on different parameterization methods of surrogate dynamics can be found in Table 4.7.1 in the appendix.

	SINDy-AE (full)	AD-EnKF (full)	ROAD-EnKF (full)	AD-EnKF (partial)	ROAD-EnKF (partial)
RMSE-r	0.1433	0.0102	0.0044	0.0122	0.0081
RMSE-f(30)	4.4579	0.0212	0.0096	0.0228	0.0160
RMSE-f(150)	4.4906	0.0763	0.0514	0.0724	0.0581
Log-likelihood	–	6.40×10^4	6.60×10^4	3.24×10^4	3.27×10^4
Training time (per epoch)	11.78s	26.75s	12.10s	27.08s	12.20s
Test time	2.78s	11.54s	4.21s	7.76s	3.24s

Table 4.5.4: Performance metrics for different algorithms at convergence. (Burgers example, §4.5.2.)

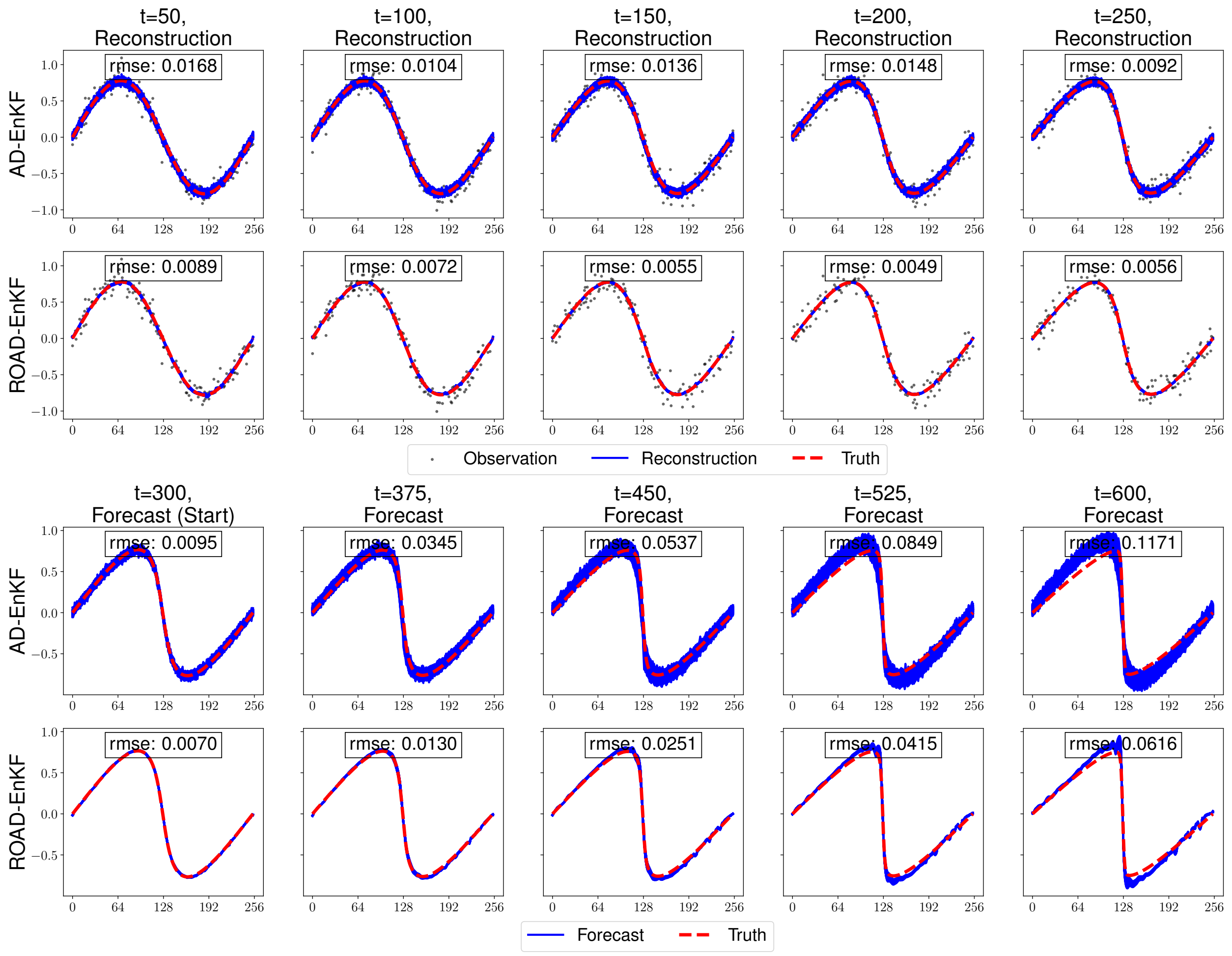
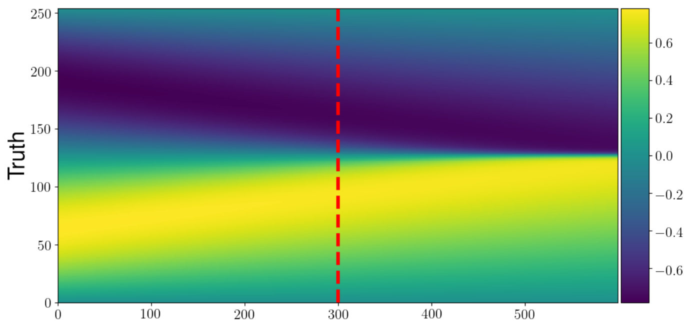
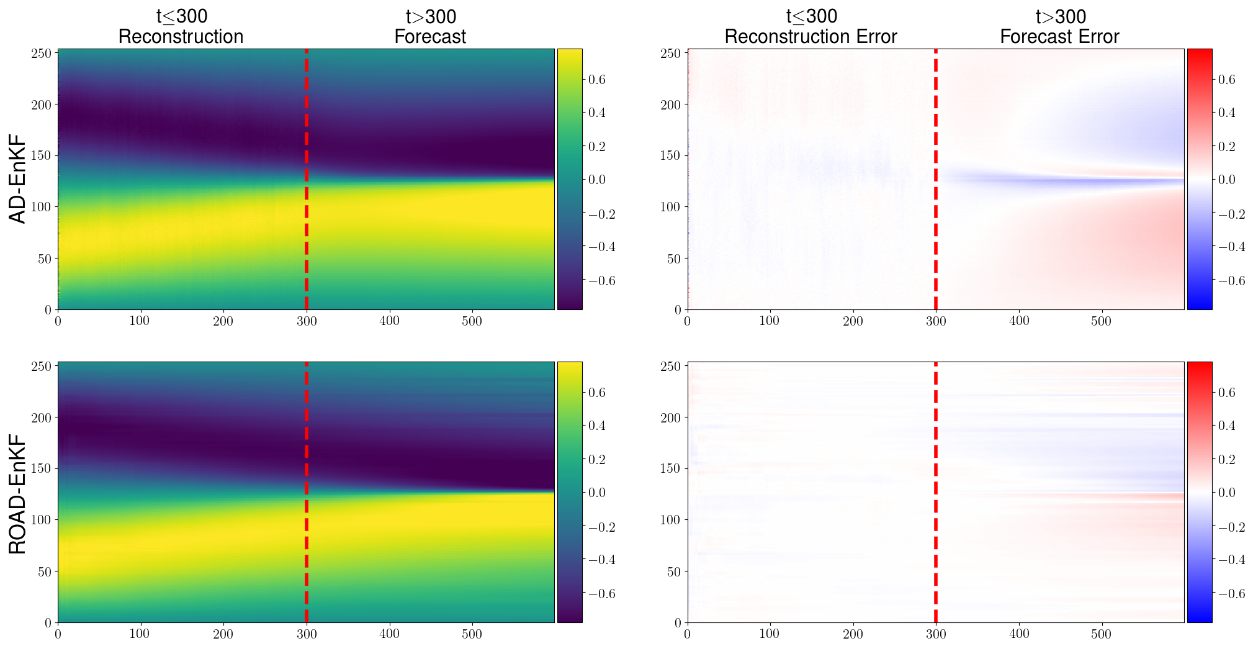


Figure 4.7: State reconstruction (upper half) and forecast (lower half) performance with partial observation ($d_u = 256$, $d_y = 128$) on the Burgers example in §4.5.2. For each method, the reconstructed states u_t (blue) for a single test sequence are plotted for $t = 50, 100, 150, 200, 250$ (column), and the forecasted states (blue) for a single test sequence are plotted for $t = 300$ (start of forecast), $375, 450, 525, 600$ (column). The true values of the 256-dimensional states are plotted in red dashed lines, along with the noisy observations in black dots. Both AD-EnKF and ROAD-EnKF perform probabilistic state reconstructions and forecast through particles (all plotted in blue). The reconstruction/forecast RMSEs are computed for each plot.



(a) Ground truth.



(b) Reconstruction and forecast.

Figure 4.8: Contour plot of state reconstruction and forecast output with partial observation ($d_u = 256$, $d_y = 128$) on the Burgers example in §4.5.2, as well as the ground truth (top). For each method (row), the reconstructed and forecasted states (left column) for a single test sequence are plotted, for each state dimension (y-axis) and time (x-axis). The error compared to the ground truth are plotted in the right column. For both AD-EnKF and ROAD-EnKF we use particle means as point estimates.

	ROAD-EnKF								AD-EnKF
	$d_z = 1$	$d_z = 2$	$d_z = 4$	$d_z = 10$	$d_z = 20$	$d_z = 40$	$d_z = 120$	$d_z = 240$	
RMSE-r	0.2293	0.0316	0.0035	0.0058	0.0039	0.0044	0.0048	0.0059	0.0102
RMSE-f(30)	0.2593	0.0761	0.0165	0.0108	0.0112	0.0096	0.0100	0.0103	0.0212
RMSE-f(150)	0.2690	0.1827	0.1313	0.0501	0.0607	0.0514	0.0373	0.0382	0.0763
Log-likelihood ($\times 10^4$)	-14.4	6.03	6.63	6.59	6.61	6.60	6.61	6.63	6.40
Training time (per epoch)	11.70s	11.78s	11.79s	11.98s	11.98s	12.10s	12.44s	13.40s	26.75s
Test time	3.36s	3.80s	4.13s	4.14s	4.08s	4.21s	4.53s	4.90s	11.54s

Table 4.5.5: Performance metrics for ROAD-EnKF at convergence with full observation ($d_u = d_y = 256$) and different latent space dimension d_z. (Burgers example, §4.5.2.)

4.5.3 Kuramoto-Sivashinsky Equation

In this subsection, we consider the Kuramoto-Sivashinsky (KS) equation for $u(x, s)$, where u is a function of the spatial variable $x \in [0, \mathsf{L}]$ and continuous-time variable $s > 0$:

$$\begin{aligned} &\frac{\partial u}{\partial s} = -\nu \frac{\partial^4 u}{\partial x^4} - \frac{\partial^2 u}{\partial x^2} - u\frac{\partial u}{\partial x}, \\ &u(0, s) = u(\mathsf{L}, s) = 0, \\ &\frac{\partial u}{\partial x}(0, s) = \frac{\partial u}{\partial x}(\mathsf{L}, s) = 0, \\ &u(x, 0) = u_0(x), \end{aligned} \tag{4.5.8}$$

Here ν is the viscosity parameter, and we set $\nu = 0.05$, $\mathsf{L} = 2$. We impose Dirichlet and Neumann boundary conditions to ensure ergodicity of the system [Blonigan and Wang, 2014]. The KS equation was originally introduced by Kuramoto and Sivashinsky to model turbulence of reaction-diffusion systems [Kuramoto and Tsuzuki, 1976] and propagation of flame [Sivashinsky, 1977]. Equation Eq. (4.5.8) is discretized on $[0, \mathsf{L}]$ with equally-spaced grid points $0 = x_1 < x_2 < \cdots < x_M = \mathsf{L}$, using a second-order finite difference method.

Setting $\Delta x := x_i - x_{i-1} = \frac{\mathsf{L}}{M-1}$, we obtain the following ODE system:

$$
\begin{aligned}
&\frac{\partial u^{(i)}}{\partial s} = -\nu \frac{u^{(i-2)} - 4u^{(i-1)} + 6u^{(i)} - 4u^{(i+2)} + u^{(i+2)}}{\Delta x^4} - \frac{u^{(i+1)} - 2u^{(i)} + u^{(i-1)}}{\Delta x^2} - \frac{\left(u^{(i+1)}\right)^2 - \left(u^{(i-1)}\right)^2}{4\Delta x}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 2, \ldots, d_u - 1, \\
&u^{(1)}(s) = u^{(d_u)}(s) = 0, \\
&u^{(0)}(s) = u^{(2)}(s), u^{(d_u+1)}(s) = u^{(d_u-1)}(s), \\
&u^{(i)}(0) = u_0(i\Delta x).
\end{aligned}
\tag{4.5.9}
$$

The discretization method follows [Wan and Sapsis, 2017]. Here $u^{(i)}(s)$ is an approximation of $u(i\Delta x, s)$, the value of u at the i-th spatial node and time s. Two ghost nodes $u^{(0)}$ and $u^{(d_u+1)}$ are added to account for Neumann boundary conditions, and are not regarded as part of the state. Equation Equation (4.5.9) defines a flow map $F : u(s) \mapsto u(s + \Delta_s)$ for state variable u with $d_u = M$, which we refer to as the true state dynamics model. We assume there is no noise in the dynamics, i.e., $Q = 0$.

Similar to §4.5.1, we consider two cases: full observation with $d_u = d_y = 256$ and partial observation with $d_u = 256$, $d_y = 128$ (i.e., $c = 1/2$). The initial conditions u_0 are generated at random from the attractor of the dynamical system, by simulating a long run beforehand. We generate $N_{\text{train}} = 512$ training data and $N_{\text{test}} = 20$ test data with the true state dynamics model defined through Eq. (4.5.9) with $R_t = I_{d_y}$. We set the number of observations $T = 450$ with time between observations $\Delta_s = 0.1$. We set the forecast lead time $T_f = 50$. The flow map F is integrated using the fourth-order Runge–Kutta method with a fine step size $\Delta_s/10000$. The surrogate latent dynamics map g_α is parameterized as a two-layer fully connected NN, and is integrated using a fourth-order Runge-Kutta method with step size $\Delta_s^{\text{int}} = 0.05$. The error covariance matrix S_β in the latent dynamics is parametrized using a diagonal matrix with positive diagonal elements $\beta \in \mathbb{R}^{d_z}$. The decoder D_γ is parameterized as an FND, discussed in §4.4.1. Details of the network hyperparameters are listed in Table 4.5.2. The latent space dimension for ROAD-EnKF is set to $d_z = 40$. The ensemble size for both AD-EnKF and ROAD-EnKF is set to $N = 100$.

In Table 4.5.6 we list the performance metrics of AD-EnKF and ROAD-EnKF with full and partial observation. SINDy-AE is not listed here as we find it unable to capture the dynamics for any choice of latent space dimension. The state reconstruction and forecast performance on a single instance of test data are plotted in Fig. 4.9 (snapshots), and Fig. 4.10 (contour plot, ROAD-EnKF) for the partial observation case. Corresponding plots with full observation are shown in Figures 4.13 and 4.14 in the appendix. We find that ROAD-EnKF is able to reconstruct the states with lower RMSE than AD-EnKF in both full observation and partial observation cases. Both methods can produce meaningful forecast multiple steps forward into the future. ROAD-EnKF achieves a higher forecast RMSE than AD-EnKF in full observation case, while having a lower forecast RMSE in partial observation case. Although ROAD-EnKF does not consistently have a better forecast performance than AD-EnKF due to the difficulty of finding a reduced-order representation for the highly chaotic system, we find that its performance is not much impacted by partial observation. Moreover, it is two times more efficient than AD-EnKF in both training and testing, due to the times saved for simulating a cheaper surrogate model and running the EnKF algorithm in a lower dimensional space. Notice in Fig. 4.10(b) and Fig. 4.14(b) that, although the predictive means of all particles are 'smoothed' when passing a certain time threshold, each particle individually produces nontrivial forecasts for a larger number of time steps into the future, thus illustrating the variability of particle forecasts and the stochastic nature of state reconstruction and forecast in our ROAD-EnKF framework.

4.6 Conclusions and Future Directions

This paper introduced a computational framework to reconstruct and forecast a partially observed state that evolves according to an unknown or expensive-to-simulate dynamical

	AD-EnKF (full)	ROAD-EnKF (full)	AD-EnKF (partial)	ROAD-EnKF (partial)
RMSE-r	0.4658	0.3552	0.4686	0.3589
RMSE-f(1)	0.5137	0.5626	0.6231	0.5644
RMSE-f(5)	1.0910	1.2734	1.4669	1.3780
Log-likelihood	-1.89×10^6	-1.88×10^6	-9.33×10^5	-9.07×10^5
Training time (per epoch)	28.92s	12.53s	28.72s	12.61s
Test time	12.35s	5.11s	6.22s	4.39s

Table 4.5.6: Performance metrics for different algorithms at convergence. (KS example, §4.5.3.)

system. Our ROAD-EnKFs use an EnKF algorithm to estimate by maximum likelihood a surrogate model for the dynamics in a latent space, as well as a decoder from latent space to state space. Our numerical experiments demonstrate the computational advantage of co-learning an inexpensive surrogate model in latent space together with a decoder, rather than a more expensive-to-simulate dynamics in state space.

The proposed computational framework accommodates partial observation of the state, does not require time derivative data, and enables uncertainty quantification. In addition, it provides significant algorithmic flexibility through the choice of latent space, surrogate model for the latent dynamics, and decoder design. In this work, we showed that accurate and cheap reconstructions and forecasts can be obtained by choosing an inexpensive NN surrogate model, and a decoder inspired by recent ideas from operator learning. While adequate choice of NN architecture and decoder may be problem-specific, an important question for further research is to derive guidelines and physics-informed NNs that are well-suited for certain classes of problems.

4.7 Appendix

4.7.1 Improving AD-EnKF with Spectral Convolutional Layers

This appendix discusses an enhancement of the AD-EnKF algorithm [Chen et al., 2022], used for numerical comparisons in Section 4.5. AD-EnKF runs EnKF on the full-order SSM

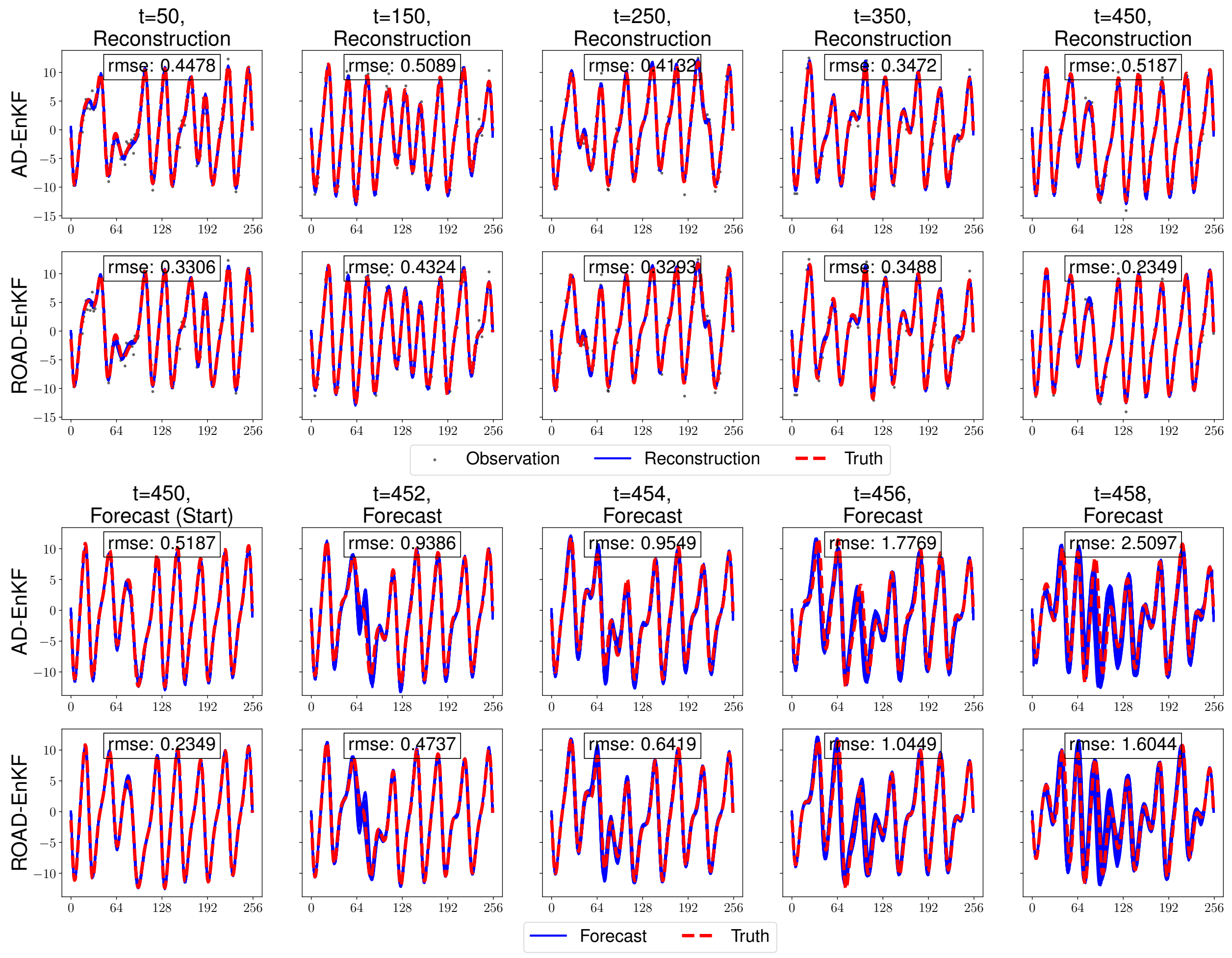
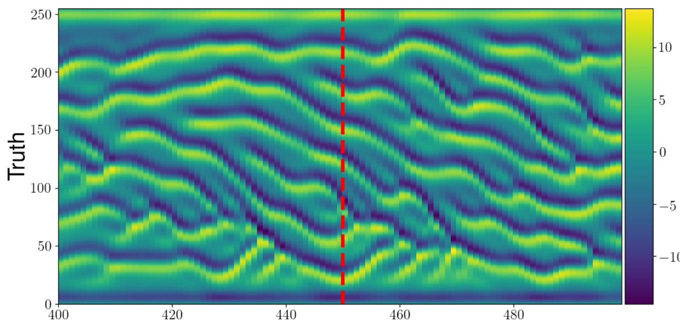
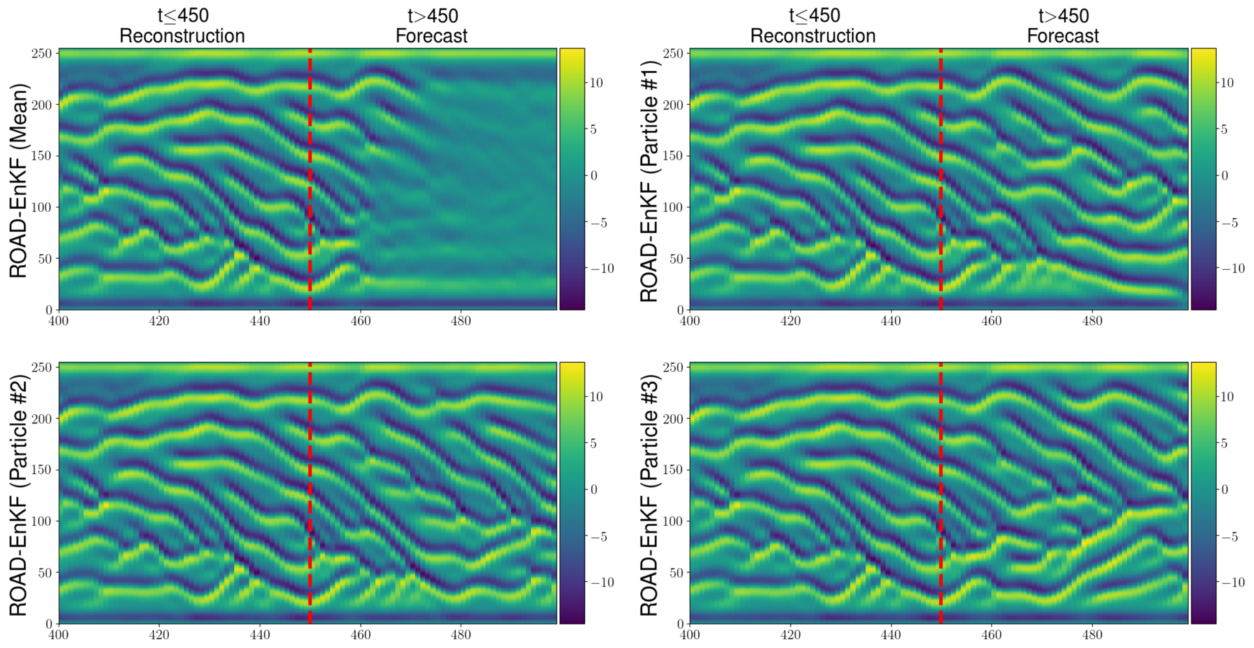


Figure 4.9: State reconstruction (upper half) and forecast (lower half) performance with partial observation ($d_u = 256$, $d_y = 128$) on the KS example in §4.5.3. For each method, the reconstructed states u_t (blue) for a single test sequence are plotted for $t = 50, 150, 250, 350, 450$ (column), and the forecasted states (blue) for a single test sequence are plotted for $t = 450$ (start of forecast), $452, 454, 456, 458$ (column). The true values of the 256-dimensional states are plotted in red dashed lines, along with the noisy observations in black dots. Both AD-EnKF and ROAD-EnKF perform probabilistic state reconstructions and forecast through particles (all plotted in blue). The reconstruction/forecast RMSEs are computed for each plot.

(4.2.1)-(4.2.3) and learns the parameter $\theta = (\alpha^\top, \beta^\top)^\top$ by auto-differentiating through a similarly defined log-likelihood objective, as in §4.3.2. A high dimension of u makes challenging the NN parameterization of F_α (resp. f_α in the ODE case) in the state dynamics model Eq. (4.2.1). In particular, the local convolutional NN used in [Chen et al., 2022] does not perform well in the high-dimensional numerical experiments considered in §4.5. We thus



(a) Ground truth.



(b) Reconstruction and forecast.

Figure 4.10: Contour plot of state reconstruction and forecast output of ROAD-EnKF with partial observation ($d_u = 256$, $d_y = 128$) on the KS example in §4.5.3, as well as the ground truth (top). The particle means of reconstructed and forecasted states for a single test sequence are plotted, for each state dimension (y-axis) and time (x-axis). The reconstructed and forecasted states of three randomly chosen particles are also plotted individually.

propose a more flexible NN parameterization of F_α (resp. f_α) using the idea of spectral convolutional layers.

We design F_α (resp. f_α) in a way similar to the Fourier Neural Decoder, but without the complex linear layer and IDFT step at the beginning. That is, we start with a state variable $u' \in \mathbb{R}^{d_u}$ as the input, iteratively apply Eq. (4.4.2) with $v_0 = u'$ to get $v_L \in \mathbb{R}^{n_L \times d_u}$, followed by a fully-connected network applied over the channel dimension to get the output $u \in \mathbb{R}^{d_u}$. The architecture is the same as Fig. 4.3(a) but we start at v_0 instead of z.

4.7.2 Additional Materials: Burgers Example

For SINDy-AE, we use a finite difference approximation computed from data $y_{1:T}$ to approximate the exact time-derivative. The latent space dimension for SINDy-AE is set to 6. Increasing it does not further enhance the performance, but increases the computational cost.

	AD-EnKF (FC, Euler)	AD-EnKF (FC, RK4)	AD-EnKF (Fourier, Euler)	AD-EnKF (Fourier, RK4)	ROAD-EnKF (FC, Euler)	ROAD-EnKF (FC, RK4)
RMSE-r	0.1023	0.0934	0.0537	0.0102	0.0045	0.0044
RMSE-f(30)	0.0999	0.0831	0.1302	0.0212	0.0100	0.0096
RMSE-f(150)	0.1971	0.1608	0.2908	0.0763	0.0664	0.0514
Log-likelihood	2.31×10^4	2.87×10^4	5.41×10^4	6.40×10^4	6.60×10^4	6.57×10^4
Training time (per epoch)	4.97s	5.80s	12.31s	26.75s	11.21s	12.10s
Test time	2.29s	2.97s	4.79s	11.54s	3.28s	4.21s

Table 4.7.1: Ablation study: AD-EnKF versus ROAD-EnKF with different NN parameterization and numerical integration methods for surrogate dynamics (FC: NN with fully-connected layers; Fourier: NN with Fourier layers; Euler: Euler method for ODE integration; RK4: fourth-order Runge Kutta method for ODE integration). Switching from RK4 to Euler method while keeping the same NN configuration gives a computational speed-up, and the speed-up is more noticeable when the NN involves Fourier layers. However, after the switch, the accuracy drops more significantly for AD-EnKF than for ROAD-EnKF. The best configuration for AD-EnKF (Fourier with RK4) still yields a lower accuracy compared to both ROAD-EnKF configurations, while taking more time to compute. (Burgers example, full observation case, §4.5.2.)

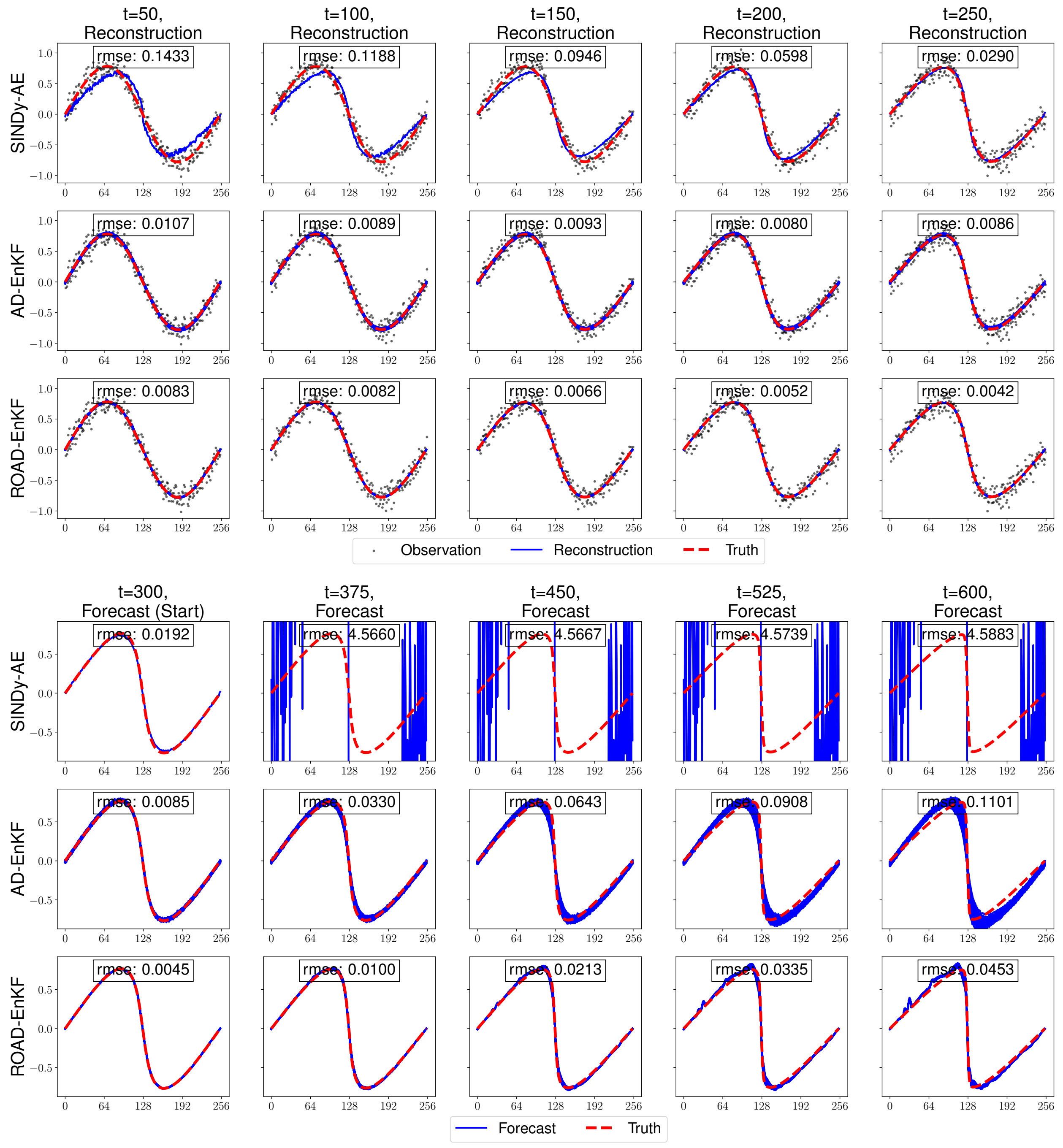
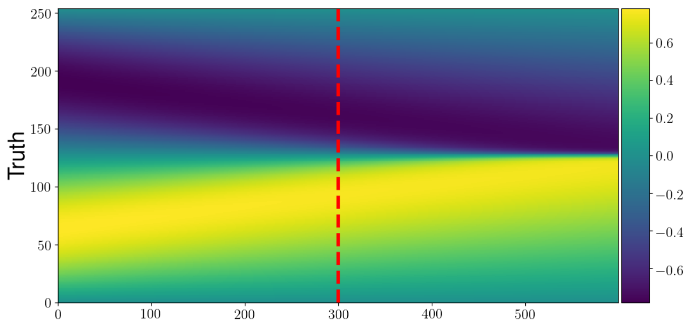
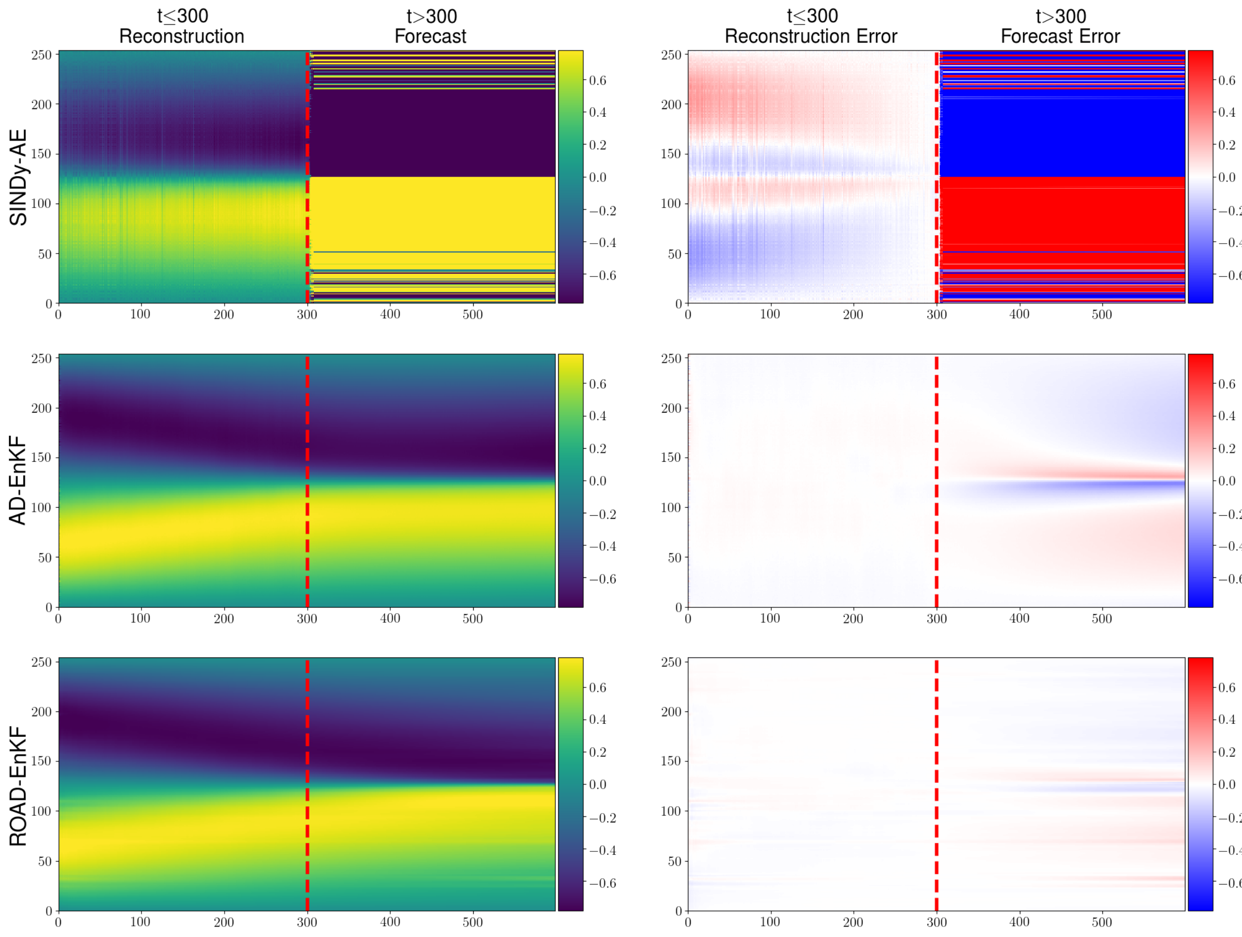


Figure 4.11: State reconstruction (upper half) and forecast (lower half) performance with full observation ($d_u = d_y = 256$) on the Burgers example in §4.5.2. For each method, the reconstructed states u_t (blue) for a single test sequence are plotted for $t = 50, 100, 150, 200, 250$ (column), and the forecasted states (blue) for a single test sequence are plotted for $t = 300$ (start of forecast), $375, 450, 525, 600$ (column). The true values of the 256-dimensional states are plotted in red dashed lines, along with the noisy observations in black dots. Both AD-EnKF and ROAD-EnKF perform probabilistic state reconstructions and forecast through particles (all plotted in blue), while SINDy-AE only provides point estimates. The reconstruction/forecast RMSEs are computed for each plot.



(a) Ground truth.



(b) Reconstruction and forecast.

Figure 4.12: Contour plot of state reconstruction and forecast output with full observation ($d_u = d_y = 256$) on the Burgers example in §4.5.2, as well as the ground truth (top). For each method (row), the reconstructed and forecasted states (left column) for a single test sequence are plotted, for each state dimension (y-axis) and time (x-axis). The error compared to the ground truth are plotted in the right column. For both AD-EnKF and ROAD-EnKF we use particle means as point estimates.

4.7.3 Additional Materials: Kuramoto-Sivashinky Example

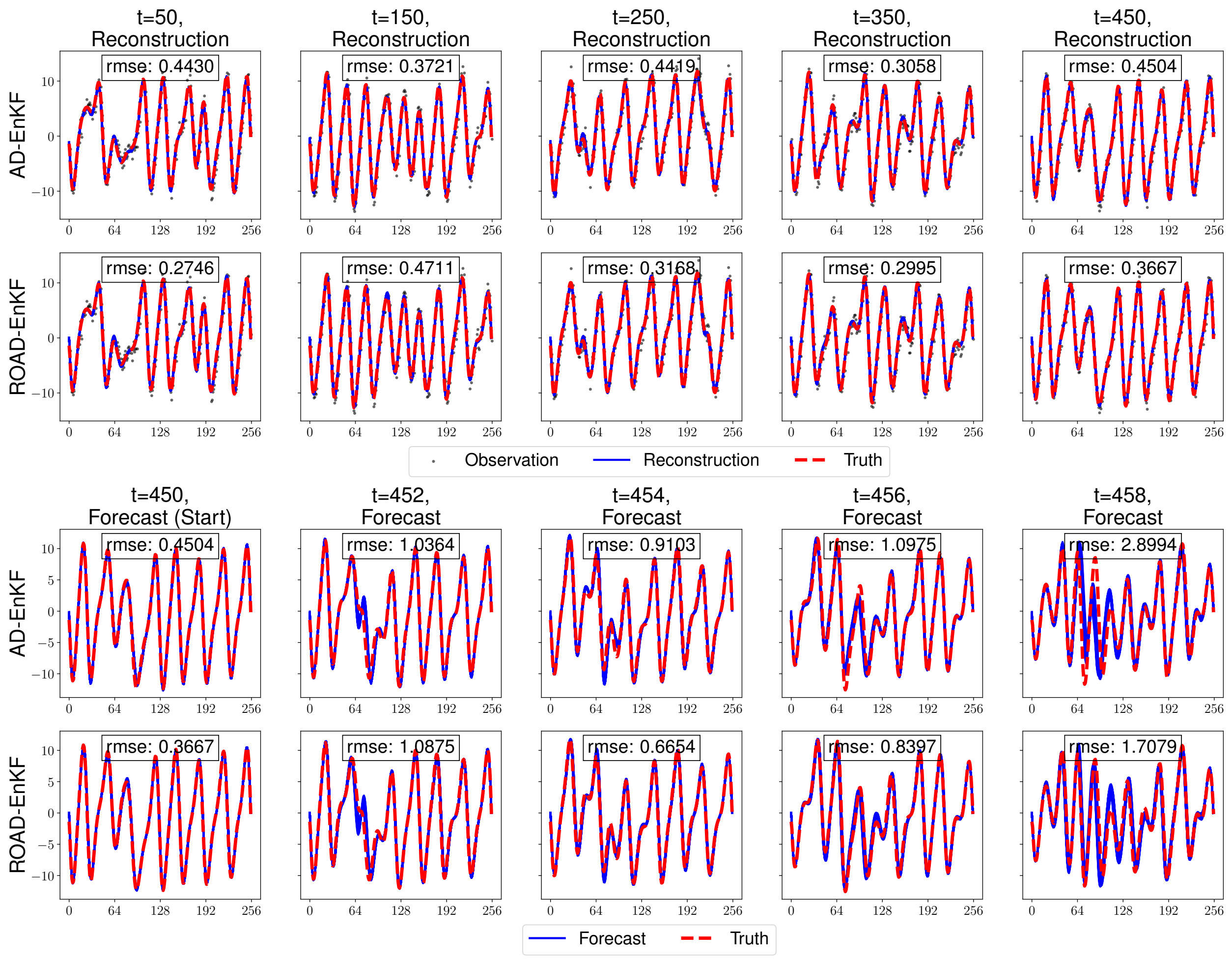
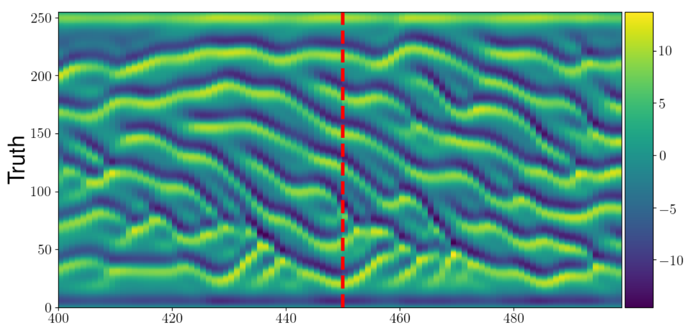
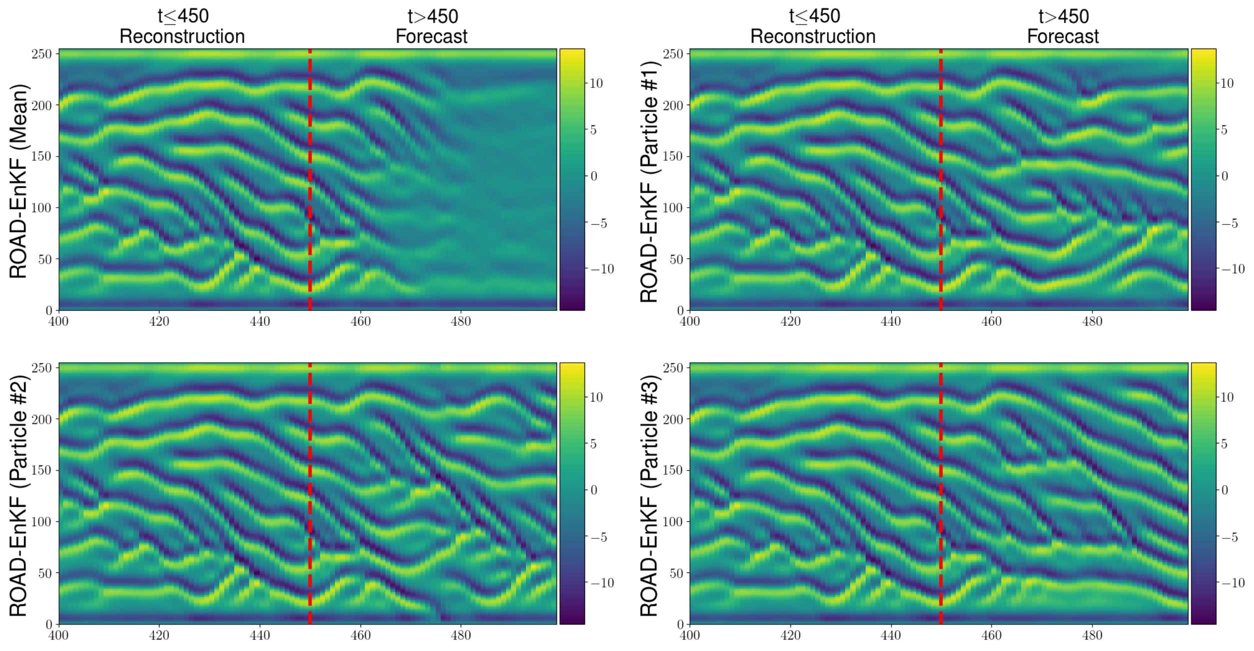


Figure 4.13: State reconstruction (upper half) and forecast (lower half) performance with full observation ($d_u = d_y = 256$) on the KS example in §4.5.3. For each method, the reconstructed states u_t (blue) are plotted for $t = 50, 150, 250, 350, 450$ (column), and the forecasted states (blue) are plotted for $t = 450$ (start of forecast), $452, 454, 456, 458$ (column). The true values of the 256-dimensional states are plotted in red dashed lines, along with the noisy observations in black dots. Both AD-EnKF and ROAD-EnKF perform probabilistic state reconstructions and forecast through particles (all plotted in blue). The reconstruction/forecast RMSEs are computed for each plot.



(a) Ground truth.



(b) Reconstruction and forecast.

Figure 4.14: Contour plot of state reconstruction and forecast output of ROAD-EnKF with full observation ($d_u = d_y = 256$) on the KS example in §4.5.3, as well as the ground truth (top). The particle means of reconstructed and forecasted states are plotted, for each state dimension (y-axis) and time (x-axis). The individual reconstructed and forecasted states of three randomly chosen particles are also plotted.

CHAPTER 5
ITERATIVE ENSEMBLE KALMAN METHODS: A UNIFIED PERSPECTIVE WITH SOME NEW VARIANTS

5.1 Introduction

This paper provides an accessible introduction to the derivation and foundations of iterative ensemble Kalman methods, a family of derivative-free algorithms for parameter reconstruction and other related tasks. The overarching theme behind these methods is to iteratively update via Kalman-type formulae an ensemble of candidate reconstructions, aiming to bring the ensemble closer to the unknown parameter with each iteration. The ensemble Kalman updates approximate derivative-based nonlinear least-squares optimization schemes without requiring gradient evaluations. Our presentation emphasizes that iterative ensemble Kalman methods can be naturally classified in terms of the nonlinear least-squares objective they seek to minimize and the derivative-based optimization scheme they approximate through the ensemble. This perspective allows us to identify three subfamilies of iterative ensemble Kalman methods, creating unity into the growing literature on this subject. Our work also emphasizes two principles for the derivation and analysis of iterative ensemble Kalman methods: statistical linearization and continuum limits. Following these principles we introduce new iterative ensemble Kalman methods that show promising numerical performance in Bayesian inverse problems, data assimilation and machine learning tasks.

We consider the application of iterative ensemble Kalman methods to the problem of reconstructing an unknown $u \in \mathbb{R}^d$ from corrupt data $y \in \mathbb{R}^k$ related by

$$y = h(u) + \eta, \tag{5.1.1}$$

where η represents measurement or model error and h is a given map. A wide range of inverse problems, data assimilation and machine learning tasks can be cast into the framework (5.1.1). In these applications the unknown u may represent, for instance, an input parameter of a differential equation, the current state of a time-evolving signal and a regressor, respectively. Ensemble Kalman methods were first introduced as filtering schemes for sequential data assimilation [Evensen, 2009; Evans and Leeuwen, 1996; Majda and Harlim, 2012; Reich and Cotter, 2013; Sanz-Alonso et al., 2018] to reduce the computational cost of the Kalman filter [Kalman, 1960]. Their use for state and parameter estimation and inverse problems was further developed in [Anderson, 2001; Lorentzen et al., 2001; Nœvdal et al., 2002; Skjervheim et al., 2011]. The idea of *iterating* these methods was considered in [Chen and Oliver, 2012; Emerick and Reynolds, 2013]. Iterative ensemble Kalman methods are now popular in inverse problems and data assimilation; they have also shown some potential in machine learning applications [Haber et al., 2018; Guth et al., 2022; Kovachki and Stuart, 2019].

Starting from an initial ensemble $\{u_0^{(n)}\}_{n=1}^N$, iterative ensemble Kalman methods use various ensemble-based empirical means and covariances to update

$$\{u_i^{(n)}\}_{n=1}^N \to \{u_{i+1}^{(n)}\}_{n=1}^N, \tag{5.1.2}$$

until a stopping criteria is satisfied; the unknown parameter u is reconstructed by the mean of the final ensemble. The idea is analogous to classical Kalman methods and optimization schemes which, starting with a *single* initialization u_0 use evaluations of derivatives of h to iteratively update $u_i \to u_{i+1}$ until a stopping criteria is met. The initial ensemble is viewed as an input to the algorithm, obtained in a problem-dependent fashion. In Bayesian inverse problems and machine learning it may be obtained by sampling a prior, while in data assimilation the initial ensemble may be a given collection of particles that approximates the prediction distribution. In either case, it is useful to view the initial ensemble as a sample from a probability distribution. It is important to note that there is no time variable involved in the reconstruction task (5.1.1); however, we will often think of the iteration index $i \in \mathbb{N}$ in (5.1.2) as an artificial time index, since this allows us to interpret the evolution of iterates as arising from discretization of differential equations, and thereby to gain theoretical understanding.

There are two main computational benefits in updating an *ensemble* of candidate reconstructions rather than a single estimate. First, the ensemble update can be performed without evaluating derivatives of h, effectively approximating them using *statistical linearization*. This is important in applications where computing derivatives of h is expensive, or where the map h needs to be treated as a black-box. Second, the use of empirical rather than model covariances can significantly reduce the computational cost whenever the ensemble size N is smaller than the dimension d of the unknown parameter u. A further advantage of the ensemble approach is that, for problems that are not strongly nonlinear, the spread of the ensemble may contain meaningful information on the uncertainty in the reconstruction.

5.1.1 Overview: Three subfamilies

This paper identifies, compares and further develops three subfamilies of iterative ensemble Kalman methods to implement the ensemble update (5.1.2). Each subfamily employs a different Kalman-type formulae, determined by a choice of objective to minimize and a derivative-based optimization scheme to approximate with the ensemble. All three approaches impose some form of regularization, either explicitly through the choice of the objective, or implicitly through the choice of the optimization scheme. Incorporating regularization is essential in parameter reconstruction problems encountered in applications, which are typically under-determined or ill-posed [Lu and Pereverzev, 2011; Sanz-Alonso et al., 2018].

The first subfamily considers a *Tikhonov-Phillips* objective associated with the parameter reconstruction problem (5.1.2), given by

$$\mathsf{J}_{\text{TP}}(u) := \frac{1}{2}|y - h(u)|_R^2 + \frac{1}{2}|u - m|_P^2, \tag{5.1.3}$$

where R and P are symmetric positive definite matrices that model, respectively, the data measurement precision and the level of regularization —incorporated explicitly through the choice of objective— and m represents a background estimate of u. Here and throughout this paper we use the notation $|v|_A^2 := |A^{-1/2}v|^2 = v^T A^{-1} v$ for symmetric positive definite A and vector v. The ensemble is used to approximate a Gauss-Newton method applied to the Tikhonov-Phillips objective J_{TP}. Algorithms in this subfamily were first introduced in geophysical data assimilation [Aanonsen et al., 2009; Chen and Oliver, 2012; Emerick and Reynolds, 2013; Gu and Oliver, 2007; Li and Reynolds, 2007; Reynolds et al., 2006] and were inspired by iterative, derivative-based, extended Kalman filters [Bell, 1994; Jazwinski, 2007]. Extensions to more challenging problems with strongly nonlinear dynamics are considered in [Sakov et al., 2012; Ungarala, 2012]. In this paper we will use a new Iterative Ensemble Kalman Filter (IEKF) method as a prototypical example of an algorithm that belongs to this subfamily.

The second subfamily considers the *data-misfit* objective

$$\mathsf{J}_{\text{DM}}(u) := \frac{1}{2}|y - h(u)|_R^2. \tag{5.1.4}$$

When the parameter reconstruction problem is ill-posed, minimizing J_{DM} leads to unstable reconstructions. For this reason, iterative ensemble Kalman methods in this subfamily are complemented with a Levenberg-Marquardt optimization scheme that implicitly incorporates regularization. The ensemble is used to approximate a regularizing Levenberg-Marquardt optimization algorithm to minimize J_{DM}. Algorithms in this subfamily were introduced in the applied mathematics literature [Iglesias, 2016; Iglesias et al., 2013] building on ideas from

Objective	Optimization	Derivative Method	Ensemble Method	New Variant
J_{TP}	GN	IExKF (5.2.1)	IEKF (5.3.1)	IEKF-SL (5.4.1)
J_{DM}	LM	LM-DM (5.2.2)	EKI (5.3.2)	EKI-SL (5.4.2)
J_{TP}	LM	LM-TP (5.2.3)	TEKI (5.3.3)	TEKI-SL (5.4.3)

Table 5.1.1: Roadmap to the algorithms considered in this paper. We use the abbreviations GN and LM for Gauss-Newton and Levenberg-Marquardt. The numbers in parenthesis represent the subsection in which each algorithm is introduced.

classical inverse problems [Hanke, 1997]. Recent theoretical work has focused on developing continuous-time and mean-field limits, as well as various convergence results [Blömker et al., 2019; Bloömker et al., 2018; Chada and Tong, 2022; Herty and Visconti, 2018; Kovachki and Stuart, 2019; Schillings and Stuart, 2017]. Methodological extensions based on Bayesian hierarchical techniques were introduced in [Chada, 2018; Chada et al., 2017] and the incorporation of constraints has been investigated in [Albers et al., 2019; Chada et al., 2019]. In this paper we will use the Ensemble Kalman Inversion (EKI) method [Iglesias et al., 2013] as a prototypical example of an algorithm that belongs to this subfamily.

The third subfamily, which has emerged more recently, combines explicit regularization through the Tikhonov-Phillips objective and an implicitly regularizing optimization scheme [Chada and Tong, 2022; Chada et al., 2020]. Precisely, a Levenberg-Marquardt scheme is approximated through the ensemble in order to minimize the Tikhonov-Phillips objective J_{TP}. In this paper we will use the Tikhonov ensemble Kalman inversion method (TEKI) [Chada et al., 2020] as a prototypical example.

To conclude this overview we note that while in this paper we will only consider least-squares objectives, iterative ensemble Kalman methods that use other regularizers have been recently proposed [Kovachki and Stuart, 2019; Lee, 2021; Schneider et al., 2022]. As well as this, we will restrict our attention to ensemble methods and their similarities with derivative-based methods. Iterative variants of other data assimilation methods such as 3DVAR and 4DVAR may be of interest [Lorenc, 1986; Mandel et al., 2013; Sanz-Alonso et al., 2018], but are outside the scope of this paper.

5.1.2 Statistical linearization, continuum limits and new variants

Each subfamily of iterative ensemble Kalman methods stems from a derivative-based optimization scheme. However, there is substantial freedom as to how to use the ensemble to approximate a derivative-based method. We will focus on randomized-maximum likelihood implementations [Gu and Oliver, 2007; Kelly and Stuart, 2014; Sanz-Alonso et al., 2018], rather than square-root or ensemble adjustment approaches [Anderson, 2001; Tippett et al., 2003; Grooms, 2020]. Two principles will guide our derivation and analysis of ensemble methods: the use of statistical linearization [Ungarala, 2012] and their connection with gradient descent methods through the study of continuum limits [Schillings and Stuart, 2017].

The idea behind statistical linearization is to approximate the gradient of h using pairs $\{(u_i^{(n)}, h(u_i^{(n)}))\}_{n=1}^N$ in such a way that if h is linear and the ensemble size N is sufficiently large, the approximation is exact. As we shall see, this idea tacitly underlies the derivation of all the ensemble methods considered in this paper, and will be explicitly employed in our derivation of new variants. Statistical linearization has also been used within Unscented Kalman filters, see e.g. [Ungarala, 2012].

Differential equations have long been important in developing and understanding optimization schemes [Nemirovskij and Yudin, 1983], and investigating the connections between differential equations and optimization is still an active area of research [Shi et al., 2021; Su et al., 2014; Wibisono et al., 2016]. In the context of iterative ensemble methods, continuum limit analyses arise from considering small length-steps and have been developed primarily in the context of EKI-type algorithms [Blömker et al., 2019; Schillings and Stuart, 2018]. While the derivative-based algorithms that motivate the ensemble methods result in an ODE continuum limit, the ensemble versions lead to a system of SDEs. Continuum limit analyses are useful in at least three ways. First, they unveil the gradient structure of the optimization schemes. Second, viewing optimization schemes as arising from discretizations of SDEs lends itself to design of algorithms that are easy to tune: the length-step is chosen to be small and the algorithms are run until statistical equilibrium is reached. Third, a simple linear-case analysis of the SDEs may be used to develop new algorithms that satisfy certain desirable properties. Our new iterative ensemble Kalman methods will be designed following these observations.

While our work advocates the study of continuum limits as a useful tool to design ensemble methods, continuum limits cannot fully capture the full richness and flexibility of discrete-based implementable algorithms, since different algorithms may result in the same SDE continuum limit. This insight suggests that it is not only the study of differential equations, but also their *discretizations*, that may contribute to the design of iterative ensemble Kalman algorithms.

5.1.3 Main contributions and outline

In addition to providing a unified perspective of the existing literature, this paper contains several original contributions. We highlight some of them in the following outline and refer to Table 5.1.1 for a summary of the algorithms considered in this paper.

- In Section 5.2 we review three iterative derivative-based methods for nonlinear least-squares optimization. The ensemble-based algorithms studied in subsequent sections can be interpreted as ensemble-based approximations of the derivative-based methods described in this section. We also derive informally ODE continuum limits for each method, which unveils their gradient flow structure.

- In Section 5.3 we describe the idea of statistical linearization. We review three subfamilies of iterative ensemble methods, each of which has an update formula analogous to one of the derivate-based methods in Section 5.2. We analyze the methods when $h(u) = Hu$ is linear by formally deriving SDE continuum limits that unveil their gradient structure. A novelty in this section is the introduction of the IEKF method, which is similar to, but different from, the iterative ensemble method introduced in [Ungarala, 2012].

- The material in Section 5.4 is novel to the best of our knowledge. We introduce new variants of the iterative ensemble Kalman methods discussed in Section 5.3 and formally derive their SDE continuum limit. We analyze the resulting SDEs when $h(u) = Hu$ is linear. The proposed methods are designed to ensure that (i) no parameter tuning or careful stopping criteria are needed; and (ii) the ensemble covariance contains meaningful information of the uncertainty in the reconstruction in the linear case, avoiding the ensemble collapse of some existing methods.
- In Section 5.5 we include an in-depth empirical comparison of the performance of the iterative ensemble Kalman methods discussed in Sections 5.3 and 5.4. We consider four problem settings motivated by applications in Bayesian inverse problems, data assimilation and machine learning. Our results illustrate the different behavior of some methods in small noise regimes and the benefits of avoiding ensemble collapse.
- Section 5.6 concludes and suggests some open directions for further research.

5.2 Derivative-based optimization for nonlinear least-squares

In this section we review the Gauss-Newton and Levenberg-Marquardt methods, two derivative-based approaches for optimization of nonlinear least-squares objectives of the form

$$\mathsf{J}(u) = \frac{1}{2}|r(u)|^2. \tag{5.2.1}$$

We derive closed formulae for the Gauss-Newton method applied to the Tikhonov-Phillips objective J_{TP}, as well as for the Levenberg-Marquardt method applied to the data-misfit objective J_{DM} and the Tikhonov-Phillips objective J_{TP}. These formulae are the basis for the ensemble, derivative-free methods considered in the next section.

As we shall see, the search directions of Gauss-Newton and Levenberg-Marquardt methods are found by minimizing a linearization of the least-squares objective. It is thus instructive to consider first linear least-squares optimization before delving into the nonlinear setting. The following well-known result, that we will use extensively, characterizes the minimizer μ of the Tikhonov-Phillips objective J_{TP} in the case of linear $h(u) = Hu$.

Lemma 5.2.1. *It holds that*

$$\frac{1}{2}|y - Hu|_R^2 + \frac{1}{2}|u - m|_P^2 = \frac{1}{2}|u - \mu|_C^2 + \beta, \tag{5.2.2}$$

where β does not depend on u, and

$$C^{-1} = H^T R^{-1} H + P^{-1}, \tag{5.2.3}$$

$$C^{-1}\mu = H^T R^{-1} y + P^{-1} m. \tag{5.2.4}$$

Equivalently,

$$\mu = m + K(y - Hm), \tag{5.2.5}$$

$$C = (I - KH)P, \tag{5.2.6}$$

where K is the Kalman gain matrix given by

$$K = PH^T(HPH^T + R)^{-1} = CH^T R^{-1}. \tag{5.2.7}$$

Proof. The formulae (5.2.3) and (5.2.4) follows by matching linear and quadratic coefficients in u between

$$\frac{1}{2}|u - \mu|_C^2 \qquad \text{and} \qquad \frac{1}{2}|u - m|_P^2 + \frac{1}{2}|y - h(u)|_R^2. \tag{5.2.8}$$

The formulae (5.2.5) and (5.2.6) as well as the equivalent expressions for the Kalman gain K in Equation (5.2.7) can be obtained using the matrix inversion lemma [Sanz-Alonso et al., 2018]. $\square$

Bayesian Interpretation Lemma 5.2.1 has a natural statistical interpretation. Consider a statistical model defined by likelihood $y|u \sim \mathcal{N}\big(Hu, R\big)$ and prior $u \sim \mathcal{N}(m, P)$. Then Equation (5.2.2) shows that the posterior distribution is Gaussian, $u|y \sim \mathcal{N}(\mu, C)$, and Equations (5.2.3)-(5.2.4) characterize the posterior mean and precision (inverse covariance). We interpret Equation (5.2.5) as providing a closed formula for the posterior *mode*, known as the *maximum a posteriori* (MAP) estimator.

More generally, the generative model

$$\begin{aligned} u &\sim \mathcal{N}(m, P), \\ y|u &\sim \mathcal{N}(h(u), R), \end{aligned} \tag{5.2.9}$$

gives rise to a posterior distribution on $u|y$ with density proportional to $\exp\big(-\mathsf{J}_{\text{TP}}$
$(u)\big)$. Thus, minimization of $\mathsf{J}_{\text{TP}}(u)$ corresponds to maximizing the posterior density under the model (5.2.9).

5.2.1 Gauss-Newton optimization of Tikhonov-Phillips objective

In this subsection we introduce two ways of writing the Gauss-Newton update applied to the Tikhonov-Phillips objective J_{TP}. We recall that the Gauss-Newton method applied to a general least-squares objective $\mathsf{J}(u) = \frac{1}{2}|r(u)|^2$ is a line-search method which, starting from an initialization u_0, sets

$$u_{i+1} = u_i + \alpha_i v_i, \qquad i = 0, 1, \dots$$

where v_i is a search direction defined by

$$v_i = \arg\min_v \mathsf{J}_i^\ell(v), \qquad \mathsf{J}_i^\ell(v) := \frac{1}{2}|r'(u_i)v + r(u_i)|^2, \tag{5.2.10}$$

where $\alpha_i > 0$ is a length-step parameter whose choice will be discussed later. In order to apply the Gauss-Newton method to the Tikhonov-Phillips objective, we write J_{TP} in the standard nonlinear least-squares form (5.2.1). Note that

$$\begin{aligned} \mathsf{J}_{\text{TP}}(u) &= \frac{1}{2}|u-m|_P^2 + \frac{1}{2}|y-h(u)|_R^2 \\ &= \frac{1}{2}|z-g(u)|_Q^2, \end{aligned}$$

where

$$z := \begin{bmatrix} y \\ m \end{bmatrix}, \qquad g(u) := \begin{bmatrix} h(u) \\ u \end{bmatrix}, \qquad Q := \begin{bmatrix} R & 0 \\ 0 & P \end{bmatrix}. \tag{5.2.11}$$

Therefore we have

$$\mathsf{J}_{\text{TP}}(u) = \frac{1}{2}|r_{\text{TP}}(u)|^2, \qquad r_{\text{TP}}(u) := Q^{-1/2}\big(z - g(u)\big). \tag{5.2.12}$$

The following result is a direct consequence of Lemma 5.2.1.

Lemma 5.2.2 ([Bell, 1994])**.** *The Gauss-Newton method applied to the Tikhonov-Phillips objective* J_{TP} *admits the characterizations:*

$$u_{i+1} = u_i + \alpha_i C_i \Big\{ H_i^T R^{-1}\big(y - h(u_i)\big) + P^{-1}(m - u_i)\Big\}, \tag{5.2.13}$$

and

$$u_{i+1} = u_i + \alpha_i \Big\{ K_i\big(y - h(u_i)\big) + (I - K_i H_i)(m - u_i)\Big\}, \tag{5.2.14}$$

where $H_i = h'(u_i)$ and

$$K_i = PH_i^T(H_iPH_i^T + R)^{-1},$$
$$C_i = (I - K_iH_i)P.$$

Proof. The search direction v_i of Gauss-Newton for the objective J_{TP} is given by

$$\begin{aligned} v_i &= \arg\min_v \mathsf{J}^{\ell}_{\text{TP},i}(v) && (5.2.15)\\ &= \arg\min_v \frac{1}{2}\big|r'_{\text{TP}}(u_i)v + r_{\text{TP}}(u_i)\big|^2 && (5.2.16)\\ &= \arg\min_v \frac{1}{2}\big|z - g(u_i) - g'(u_i)v\big|_Q && (5.2.17)\\ &= \arg\min_v \Big\{\frac{1}{2}\big|y - h(u_i) - h'(u_i)v\big|_R^2 + \frac{1}{2}\big|v - (m - u_i)\big|_P^2\Big\}. && (5.2.18)\end{aligned}$$

Applying Lemma 5.2.1, using formulae (5.2.4) and (5.2.6), we deduce that

$$v_i = C_i\Big\{H_i^TR^{-1}\big(y - h(u_i)\big) + P^{-1}(m - u_i)\Big\},$$

which establishes the characterization (5.2.13). The equivalence between (5.2.13) and (5.2.14) follows from the identity (5.2.7), which implies that $C_iH_i^TR^{-1} = K_i$ and $C_iP^{-1} = I - K_iH_i$. □

We refer to the Gauss-Newton method with constant length-step $\alpha_i = \alpha$ applied to J_{TP} as the Iterative Extended Kalman Filter (IExKF) algorithm. IExKF was developed in the control theory literature [Jazwinski, 2007] without reference to the Gauss-Newton optimization method; the agreement between both methods was established in [Bell, 1994]. In order to compare IExKF with an ensemble-based method in Section 5.3, we summarize it here.

The next proposition shows that in the linear case, if α is set to 1, IExKF finds the

Algorithm 5.2.1 Iterative Extended Kalman Filter (IExKF)

```
1: **Input**: Initialization $u_0 = m$, length-step $\alpha$.
2: For $i = 0, 1, \dots$ do:
        Set $K_i = PH_i^T(H_iPH_i^T + R)^{-1}, \qquad H_i = h'(u_i)$.
        Set $C_i = (I - K_iH_i)P$.
        Set
            $u_{i+1} = u_i + \alpha\Big\{K_i\big(y - h(u_i)\big) + (I - K_iH_i)(m - u_i)\Big\}, \qquad (5.2.19)$
        or, equivalently,
            $u_{i+1} = u_i + \alpha C_i\Big\{H_i^T R^{-1}(y - h(u_i)) + P^{-1}(m - u_i)\Big\}, \qquad (5.2.20)$
3: **Output**: $u_1, u_2, \dots$
```

minimizer of the objective (5.1.3) in one iteration, and further iterations still agree with the minimizer.

Proposition 5.2.3. *Suppose that $h(u) = Hu$ is linear and $\alpha = 1$. Then the output of Algorithm 5.2.1 satisfies*

$$u_i = \mu, \qquad i = 1, 2, \dots$$

where μ is the minimizer of the Tikhonov-Phillips objective (5.1.3).

Proof. In the linear case we have

$$H_i = H, \qquad K_i = K = PH^T(HPH^T + R)^{-1}, \qquad i = 0, 1, \dots$$

Therefore, update (5.2.19) simplifies as

$$u_{i+1} = m + K(y - Hm), \quad i = 0, 1, \dots$$

This implies that, for all $i \geq 1$, it holds that $u_i = \mu$ with μ defined in Equation (5.2.5). $\square$

Choice of Length-Step When implementing Gauss-Newton methods, it is standard practice to perform a line search in the direction of v_i to adaptively choose the length-step α_i. For instance, a common strategy is to guarantee that the Wolfe conditions are satisfied [Dennis Jr and Schnabel, 1996; Majda and Harlim, 2012]. In this paper we will instead simply set $\alpha_i = \alpha$ for some fixed value of α, and we will follow a similar approach for all the derivative-based and ensemble-based algorithms we consider. There are two main motivations for doing so. First, it is appealing from a practical viewpoint to avoid performing a line search for ensemble-based algorithms. Second, when α is small each derivative-based algorithms we consider can be interpreted as a discretization of an ODE system, while the ensemble-based methods arise as discretizations of SDE systems. These ODEs and SDEs allow us to compare and gain transparent understanding of the gradient structure of the algorithms. They will also allow us to propose some new variants of existing ensemble Kalman methods. We next describe the continuum limit structure of IExKF.

Continuum Limit It is not hard to check that the term in brackets in the update (5.2.20)

$$H_i^T R^{-1}(y - h(u_i)) + P^{-1}(m - u_i)$$

is the negative gradient of $\mathsf{J}_{\text{TP}}(u)$, which reveals the following gradient flow structure in the limit of small length-step α :

$$\begin{aligned}\dot{u} &= C(t)\Big\{h'\big(u(t)\big)^T R^{-1}\big(y - h\big(u(t)\big)\big) + P^{-1}\big(m - u(t)\big)\Big\} \\ &= -C(t)\mathsf{J}'_{\text{TP}}\big(u(t)\big),\end{aligned} \tag{5.2.21}$$

with preconditioner

$$C(t) := \Big(h'\big(u(t)\big)^T R^{-1} h'\big(u(t)\big) + P^{-1}\Big)^{-1}.$$

We remark that in the linear case, $C(t) \equiv C$, where C is the posterior covariance given by (5.2.6), which agrees with the inverse of the Hessian of the Tikhonov Phillips objective.

5.2.2 Levenberg-Marquardt optimization of data-misfit objective

In this subsection we introduce the Levenberg-Marquardt algorithm and describe its application to the data misfit objective J_{DM}. We recall that the Levenberg-Marquardt method applied to a general least-squares objective $\mathsf{J}(u) = \frac{1}{2}|r(u)|^2$ is a trust region method which, starting from an initialization u_0, sets

$$u_{i+1} = u_i + v_i, \qquad i = 0, 1, \dots$$

where

$$v_i = \arg\min_v \mathsf{J}_i^\ell(v), \quad \text{s.t. } |v|_P^2 \le \delta_i, \qquad \mathsf{J}_i^\ell(v) := \frac{1}{2}|r'(u_i)v + r(u_i)|^2.$$

Similar to Gauss-Newton methods, the increment v_i is defined as the minimizer of a linearized objective, but now the minimization is constrained to a ball $\{|v|_P^2 \le \delta_i\}$ in which we *trust* that the objective can be replaced by its linearization. The increment can also be written as

$$v_i = \arg\min_v \mathsf{J}_i^{\text{UC}}(v),$$

where

$$\mathsf{J}_i^{\text{UC}}(v) = \mathsf{J}_i^\ell(v) + \frac{1}{2\alpha_i}|v|_P^2. \tag{5.2.22}$$

The parameter $\alpha_i > 0$ plays an analogous role to the length-step in Gauss-Newton methods. Note that the Levenberg-Marquardt increment is the unconstrained minimizer of a *regularized* objective. It is for this reason that we say that Levenberg-Marquardt provides an implicit regularization.

We next consider application of the Levenberg-Marquardt method to the data-misfit objective J_{DM}, which we write in standard nonlinear least-squares form:

$$\mathsf{J}_{\text{DM}}(u) = \frac{1}{2}|r_{\text{DM}}(u)|^2, \qquad r_{\text{DM}}(u) := R^{-1/2}\big(y - h(u)\big). \tag{5.2.23}$$

Lemma 5.2.4. *The Levenberg-Marquardt method applied to the data misfit objective* J_{DM} *admits the following characterization:*

$$u_{i+1} = u_i + K_i\Big\{y - h(u_i)\Big\}, \tag{5.2.24}$$

where

$$K_i = \alpha_i P H_i^T(\alpha_i H_i P H_i^T + R)^{-1}, \qquad H_i = h'(u_i).$$

Proof. Note that the increment v_i is defined as the unconstrained minimizer of

$$\begin{aligned} \mathsf{J}^{\text{UC}}_{\text{DM},i}(v) &= \frac{1}{2}|r'_{\text{DM}}(u_i)v + r_{\text{DM}}(u_i)|^2 + \frac{1}{2\alpha_i}|v|_P^2 \\ &= \frac{1}{2}|y - h(u_i) - h'(u_i)v|_R^2 + \frac{1}{2\alpha_i}|v|_P^2. \end{aligned} \tag{5.2.25}$$

The result follows from Lemma 5.2.1. □

Similar to the previous section, we will focus on implementations with constant length-step $\alpha_i = \alpha$, which leads to the following algorithm.

Algorithm 5.2.2 Iterative Levenberg-Marquardt with Data Misfit (ILM-DM)

```
1: **Input**: Initialization $u_0 = m$, length-step $\alpha$.
2: For $i = 0, 1, \dots$ do:
        Set $K_i = \alpha P H_i^T(\alpha H_i P H_i^T + R)^{-1}, \qquad H_i = h'(u_i)$.
        Set
            $u_{i+1} = u_i + K_i\Big\{y - h(u_i)\Big\}$.        (5.2.26)
3: **Output**: $u_1, u_2, \dots$
```

When $\alpha = 1$, the following linear-case result shows that ILM-DM, i.e. Algorithm 5.2.2, reaches the minimizer of J_{TP} in one iteration. However, in contrast to IExKF, further iterations of ILM-DM will typically worsen the optimization of J_{TP}, and start moving towards minimizers of J_{DM}.

Proposition 5.2.5. *Suppose that $h(u) = Hu$ is linear and $\alpha = 1$. Then the output of Algorithm 5.2.2 satisfies*

$$u_1 = \arg\min_u \mathsf{J}_{\text{TP}}(u),$$

where J_{TP} is the Tikhonov-Phillips objective (5.1.3).

Proof. The proof is identical to that of Proposition 5.2.3, noting that in the linear case $u_{i+1} = u_i + K(y - Hu_i)$. □

Example 5.2.6 (Convergence of ILM-DM with invertible observation map)**.** *Suppose that $H \in \mathbb{R}^{d\times d}$ is invertible and $\alpha = 1$. Then, writing*

$$u_{i+1} = (I - KH)u_i + Ky$$

and noting that $\rho(I - KH) < 1$ [Anderson and Moore, 2012], it follows that $u_i \to u^*$, where u^* is the unique solution to $y = Hu$. That is, the iterates of ILM-DM converge to the unique minimizer of the data misfit objective J_{DM}.

Choice of Length-Step When implementing Levenberg-Marquardt algorithms, the length-step parameter α_i is often chosen adaptively, based on the objective. However, similar to Section 5.2.1, we fix $\alpha_i = \alpha$ to be a small value which leads to an ODE continuum limit.

Continuum Limit We notice that, in the limit of small length-step α, update (5.2.26) can be written as

$$u_{i+1} = u_i + \alpha P H_i^T R^{-1}\Big\{y - h(u_i)\Big\}.$$

The term $H_i^T R^{-1}\Big\{y - h(u_i)\Big\}$ is the negative gradient of $\mathsf{J}_{\text{DM}}(u)$, which reveals the following gradient flow structure

$$\begin{aligned}\dot{u} &= Ph'\big(u(t)\big)^T R^{-1}\Big\{y - h\big(u(t)\big)\Big\} \\ &= -P\mathsf{J}'_{\text{DM}}(u),\end{aligned} \tag{5.2.27}$$

where the preconditioner P is interpreted as the prior covariance in the Bayesian framework.

5.2.3 Levenberg-Marquardt optimization of Tikhonov-Phillips objective

In this subsection we describe the application of the Levenberg-Marquardt algorithm to the Tikhonov-Phillips objective J_{TP}.

Lemma 5.2.7. *The Levenberg-Marquardt method applied to the Tikhonov-Phillips objective* J_{TP} *admits the following characterization:*

$$u_{i+1} = u_i + K_i\Big\{z - g(u_i)\Big\},$$

where

$$K_i = \alpha_i P G_i^T(\alpha_i G_i P G_i^T + Q)^{-1}, \qquad G_i = g'(u_i),$$

recalling that g is defined in (5.2.11)*.*

Proof. Note that the increment v_i is defined as the unconstrained minimizer of

$$\mathsf{J}^{\text{UC}}_{\text{TP},i}(v) = \mathsf{J}^{\ell}_{\text{TP},i}(v) + \frac{1}{2\alpha_i}|v|_P^2 \tag{5.2.28}$$

$$= \frac{1}{2}|z - g(u_i) - g'(u_i)v|_Q^2 + \frac{1}{2\alpha_i}|v|_P^2, \tag{5.2.29}$$

which has the same form as Equation (5.2.25) replacing y with z, h with g, and R with Q. □

Setting $\alpha_i = \alpha$ leads to the following algorithm.

Algorithm 5.2.3 Iterative Levenberg-Marquardt with Tikhonov-Phillips (ILM-TP)

```
1: **Input**: Initialization $u_0 = m$, length-step $\alpha$.
2: For $i = 0, 1, \dots$ do:

       Set $K_i = \alpha P G_i^T (\alpha G_i P G_i^T + Q)^{-1}, \qquad G_i = g'(u_i)$.
       Set
           $u_{i+1} = u_i + K_i \Big\{ z - g(u_i) \Big\}.$      (5.2.30)

3: **Output**: $u_1, u_2, \dots$
```

Proposition 5.2.8. *Suppose that $h(u) = Hu$ is linear and $\alpha = 1$. The output of Algorithm 5.2.3 satisfies*

$$u_1 = \arg\min_u \Big\{ \mathsf{J}_{\text{TP}}(u) + \frac{1}{2}|u - m|^2 \Big\}. \tag{5.2.31}$$

Proof. The result is a corollary of Proposition 5.2.5. To see this, note that $\mathsf{J}_{\text{TP}}(u) = \frac{1}{2}|z - g(u)|_Q^2$ can be viewed as a data-misfit objective with data z, forward model $g(u)$ and observation matrix Q. Then, ILM-TP can be interpreted as applying ILM-DM to J_{TP}, and the objective $\mathsf{J}_{\text{TP}}(u) + \frac{1}{2}|u - m|^2$ as its Tikhonov-Phillips regularization. □

Example 5.2.9 (Convergence of ILM-TP with linear invertible observation map.)**.** *It is again instructive to consider the case where $H \in \mathbb{R}^{d \times d}$ is invertible. Then, following the same reasoning as in Example 5.2.6 we deduce that the iterates of ILM-TP converge to the unique minimizer of* J_{TP}.

Continuum Limit In the limit of small length-step α, we can derive the gradient flow structure of update (5.2.30). This is similar to Section 5.2.2, with J_{DM} replaced by J_{TP}.

Objective	Optimization	Derivative Method	Ensemble Method
J_{TP}	GN	IExKF	IEKF
J_{DM}	LM	ILM-DM	EKI
J_{TP}	LM	ILM-TP	TEKI

Table 5.3.1: Summary of the main algorithms in Sections 5.2 and 5.3.

Precisely, the gradient flow of ILM-TP is given by

$$\begin{aligned}\dot{u} &= Pg'\big(u(t)\big)^T Q^{-1}\Big\{z - g\big(u(t)\big)\Big\} \\ &= -P\mathsf{J}_{\text{TP}}'(u).\end{aligned}$$

5.3 Ensemble-based optimization for nonlinear least-squares

In this section we review three subfamilies of iterative methods that update an ensemble $\{u_i^{(n)}\}_{n=1}^N$ employing Kalman-based formulae, where $i = 0, 1, \dots$ denotes the iteration index and N is a fixed ensemble size. Each ensemble member $u_i^{(n)}$ is updated by optimizing a (random) objective $\mathsf{J}_i^{(n)}$ defined using the current ensemble $\{u_i^{(n)}\}_{n=1}^N$ and/or the initial ensemble $\{u_0^{(n)}\}_{n=1}^N$. The optimization is performed without evaluating derivatives by invoking a *statistical linearization* of a Gauss-Newton or Levenberg-Marquardt algorithm. In analogy with the previous section, the three subfamilies of ensemble methods we consider differ in the choice of the objective and in the choice of the optimization algorithm. Table 5.3.1 summarizes the derivative and ensemble methods considered in the previous and the current section.

Given an ensemble $\{u_i^{(n)}\}_{n=1}^N$ we use the following notation for ensemble empirical means

$$m_i = \frac{1}{N}\sum_{n=1}^N u_i^{(n)}, \qquad h_i = \frac{1}{N}\sum_{n=1}^N h(u_i^{(n)}),$$

and empirical covariances

$$
\begin{aligned}
P_i^{uu} &= \frac{1}{N}\sum_{n=1}^{N}(u_i^{(n)} - m_i)(u_i^{(n)} - m_i)^T,\\
P_i^{uy} &= \frac{1}{N}\sum_{n=1}^{N}\bigl(u_i^{(n)} - m_i\bigr)\bigl(h(u_i^{(n)}) - h_i\bigr)^T,\\
P_i^{yy} &= \frac{1}{N}\sum_{n=1}^{N}\bigl(h(u_i^{(n)}) - h_i\bigr)\bigl(h(u_i^{(n)}) - h_i\bigr)^T.
\end{aligned}
$$

Two overarching themes that underlie the derivation and analysis of the ensemble methods studied in this section and the following one are the use of a statistical linearization to avoid evaluation of derivatives, and the study of continuum limits. We next introduce these two ideas.

Statistical Linearization If $h(u) = Hu$ is linear, we have

$$
P_i^{uy} = P_i^{uu} H^T,
$$

which motivates the approximation in the general nonlinear case

$$
h'(u_i^{(n)}) \approx (P_i^{uy})^T (P_i^{uu})^{-1} =: H_i, \qquad n = 1, \dots, N, \tag{5.3.1}
$$

where here and in what follows $(P_i^{uu})^{-1}$ denotes the pseudoinverse of P_i^{uu}. Notice that (5.3.1) can be regarded as a linear least-squares fit of pairs $\bigl\{\bigl(u_i^{(n)}, h(u_i^{(n)})\bigr)\bigr\}_{n=1}^{N}$ normalized around their corresponding empirical means m_i and h_i. We remark that in order for the approximation in (5.3.1) to be accurate, the ensemble size N should not be much smaller than the input dimension d.

Continuum Limit We will gain theoretical understanding by studying continuum limits. Specifically, each algorithm includes a length-step parameter $\alpha > 0$, and the evolution of the ensemble for small α can be interpreted as a discretization of an SDE system. We denote by $\{u^{(n)}(t)\}_{n=1}^N$ the sample paths of the underlying SDE. For each $1 \le n \le N$, we have $u_0^{(n)} = u^{(n)}(0)$, and we view $u_i^{(n)}$ as an approximation of $u^{(n)}(t)$ for $t = \alpha i$. Similarly as above, we define $P^{uu}(t), P^{uy}(t), P^{yy}(t)$ as the corresponding empirical covariances at time $t \ge 0$.

Remark 5.3.1. *For the subsequent algorithms we will employ random perturbations $y_i^{(n)}$ of the original data y. Randomly perturbing the data is common practice for ensemble methods to ensure the correct statistics in the large ensemble limit under linearity assumptions [Lawson and Hansen, 2004].*

5.3.1 Ensemble Gauss-Newton optimization of Tikhonov-Phillips objective

Iterative Ensemble Kalman Filter

Given an ensemble $\{u_i^{(n)}\}_{n=1}^N$, consider the following Gauss-Newton update for each n:

$$u_{i+1}^{(n)} = u_i^{(n)} + \alpha v_i^{(n)}, \tag{5.3.2}$$

where $\alpha > 0$ is the length-step, and $v_i^{(n)}$ is the minimizer of the following (linearized) Tikhonov-Phillips objective (cf. equation (5.2.18))

$$\mathsf{J}_{\text{TP},i}^{(n)}(v) = \frac{1}{2}\big|y_i^{(n)} - h(u_i^{(n)}) - H_i v\big|_R^2 + \frac{1}{2}\big|u_0^{(n)} - u_i^{(n)} - v\big|_{P_0^{uu}}^2, \qquad y_i^{(n)} \sim \mathcal{N}(y, \alpha^{-1}R). \tag{5.3.3}$$

Notice that we adopt the statistical linearization (5.3.1) in the above formulation. Applying Lemma 5.2.1, the minimizer $v_i^{(n)}$ can be calculated as

$$v_i^{(n)} = C_i\Big\{H_i^T R^{-1}\big(y_i^{(n)} - h(u_i^{(n)})\big) + (P_0^{uu})^{-1}\big(u_0^{(n)} - u_i^{(n)}\big)\Big\}, \tag{5.3.4}$$

or, in an equivalent form,

$$v_i^{(n)} = K_i\big(y_i^{(n)} - h(u_i^{(n)})\big) + (I - K_i H_i)(u_0^{(n)} - u_i^{(n)}), \tag{5.3.5}$$

where

$$\begin{aligned} C_i &= \big(H_i^T R^{-1} H_i + (P_0^{uu})^{-1}\big)^{-1}, \\ K_i &= P_0^{uu} H_i^T (H_i P_0^{uu} H_i^T + R)^{-1}. \end{aligned}$$

Combining (5.3.2) and (5.3.5) leads to the Iterative Ensemble Kalman Filter (IEKF) algorithm.

Algorithm 5.3.1 Iterative Ensemble Kalman Filter (IEKF)

```
1: Input: Initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ sampled independently from the prior, length-step $\alpha$.
2: For $i = 0, 1, \dots$ do:
       Set $K_i = P_0^{uu} H_i^T (H_i P_0^{uu} H_i^T + R)^{-1}$,    $H_i = (P_i^{uy})^T (P_i^{uu})^{-1}$.
       Draw $y_i^{(n)} \sim \mathcal{N}(y, \alpha^{-1} R)$ and set
       $u_{i+1}^{(n)} = u_i^{(n)} + \alpha\Big\{K_i\big(y_i^{(n)} - h(u_i^{(n)})\big) + (I - K_i H_i)\big(u_0^{(n)} - u_i^{(n)}\big)\Big\}$,    $1 \le n \le N$.    (5.3.6)
3: Output: Ensemble means $m_1, m_2, \dots$
```

We highlight that IEKF is a natural ensemble-based version of the derivative-based IExKF Algorithm 5.2.1 with update (5.2.19). Algorithm 5.3.1 is a slight modification of the iterative ensemble Kalman algorithm proposed in [Ungarala, 2012]. The difference is that [Ungarala, 2012] sets $H_i = (P_i^{uy})^T(P_0^{uu})^{-1}$ rather than $H_i = (P_i^{uy})^T(P_i^{uu})^{-1}$. Our modification guarantees that Algorithm 5.3.1 is well-balanced in the sense that if $\alpha = 1$, $u_0^{(n)} \sim \mathcal{N}(m, P)$ and $h(u) = Hu$ is linear, then the output of Algorithm 5.3.1 satisfies that, as $N \to \infty$,

$$m_i \to \mu, \qquad i = 1, 2, \dots$$

where μ is the minimizer of $\mathsf{J}_{\text{TP}}(u)$ given in Equation (5.1.3). This is analogous to Proposition 5.2.3 for IExKF. A detailed explanation is included in Section 5.3.1 below.

Other statistical linearizations and approximations of the Gauss-Newton scheme are possible. We next give a high-level description of the method proposed in [Reynolds et al., 2006], one of the earliest applications of iterative ensemble Kalman methods for inversion in the petroleum engineering literature. Consider the alternative characterization of the Gauss-Newton update (5.3.4). However, instead of using a different preconditioner C_i for each step, [Reynolds et al., 2006] uses a fixed preconditioner $C_* = P_0^{uu} - P_0^{uy}(R + P_0^{yy})^{-1}(P_0^{uy})^T$. Note that C_* can be viewed as an approximation of C_0 :

$$\begin{aligned} C_* &\approx P_0^{uu} - P_0^{uu}H_0^T(R + H_0P_0^{uu}H_0^T)^{-1}H_0P_0^{uu} \\ &= \big(H_0^TR^{-1}H_0 + (P_0^{uu})^{-1}\big)^{-1} = C_0. \end{aligned}$$

This leads to the following algorithm.

We note that IEKF-RZL, where RZL refers to the authors of the work [Reynolds et al., 2006], is a natural ensemble-based version of the derivative-based IExKF Algorithm 5.2.1 with update (5.2.20). We have empirically observed in a wide range of numerical experiments that Algorithm 5.3.1 is more stable than Algorithm 5.3.2, and we now give a heuristic argument for the advantage of Algorithm 5.3.1 in small noise regimes.

Algorithm 5.3.2 Iterative Ensemble Kalman Filter (IEKF-RZL)

```
1: Input: Initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ sampled independently from the prior, length-step $\alpha$.
       Set $C_* = P_0^{uu} - P_0^{uy}\big(R + P_0^{yy}\big)^{-1}(P_0^{uy})^T$.
2: For $i = 0, 1, \dots$ do:
       Set $H_i = (P_i^{uy})^T (P_i^{uu})^{-1}$
       Draw $y_i^{(n)} \sim \mathcal{N}(y, \alpha^{-1}R)$ and set
       $u_{i+1}^{(n)} = u_i^{(n)} + \alpha\, C_* \Big\{ H_i^T R^{-1}\big(y_i^{(n)} - h(u_i^{(n)})\big) + (P_0^{uu})^{-1}\big(u_0^{(n)} - u_i^{(n)}\big)\Big\}, \qquad 1 \le n \le N.$
3: Output: Ensemble means $m_1, m_2, \dots$
```

- The update formula in Algorithm 5.3.1 is motivated by the large N approximation

$$P_0^{uu} H_i^T (H_i P_0^{uu} H_i^T + R)^{-1} \approx P H_i^T (H_i P H_i^T + R)^{-1},$$

  which only requires that P_0^{uu} is a good approximation of P.

- The update formula in Algorithm 5.3.2 may be derived by invoking a large N approximation of several terms. In particular, the error arising from the approximation

$$C_* H_i^T R^{-1} \approx C_0 H_i^T R^{-1},$$

  gets amplified when R is small.

Empirical evidence of the instability of Algorithm 5.3.2 will be given in Section 5.5.1.

Analysis of IEKF

In the literature [Reynolds et al., 2006; Ungarala, 2012], the length-step α is sometimes set to be 1. Here we state a simple observation about Algorithm 5.3.1 when $\alpha = 1$ and $h(u) = Hu$ is linear. We further assume that $H_i \equiv H$ for all i. Then $K_i \equiv K_0 := P_0^{uu} H^T (H P_0^{uu} H^T + R)^{-1}$, and the update (5.3.6) can be simplified as

$$\begin{aligned} u_{i+1}^{(n)} &= u_i^{(n)} + K_0(y_i^{(n)} - Hu_i^{(n)}) + (I - K_0 H)(u_0^{(n)} - u_i^{(n)}) \\ &= u_0^{(n)} + K_0(y_i^{(n)} - Hu_0^{(n)}), \end{aligned}$$

where $y_i^{(n)} \sim \mathcal{N}(y, R)$. If $\{u_0^{(n)}\}_{n=1}^N$ are sampled independently from the prior $\mathcal{N}(m, P)$ and we let $N \to \infty$, we have $K_0 \to K = PH^T(HPH^T + R)^{-1}$ and, by the law of large numbers, for any $i \geq 1$,

$$\begin{aligned} \frac{1}{N}\sum_{n=1}^{N} u_i^{(n)} &= (I - K_0 H) \cdot \frac{1}{N}\sum_{n=1}^{N} u_0^{(n)} + K_0 \cdot \frac{1}{N}\sum_{n=1}^{N} y_i^{(n)} \\ &\to (I - KH)m + Ky, \end{aligned}$$

which is the posterior mean (and mode) in the linear setting. In other words, in the large ensemble limit, the ensemble mean recovers the posterior mean *after one iteration.* However, while the choice $\alpha = 1$ may be effective in low dimensional (nonlinear) inverse problems, we do not recommend it when the dimensionality is high, as H_i might not be a good approximation of H. Thus, we introduce Algorithms 5.3.1 and 5.3.2 with a choice of length-step α, resembling the derivative-based Gauss-Newton method discussed in Section 5.2.1. Further analysis of a new variant of Algorithm 5.3.1 will be conducted in a continuum limit setting in Section 5.4.

Remark 5.3.2. *Some remarks:*

1. *Although this will not be the focus of our paper, the IEKF Algorithm 5.3.1 (together*

with the EKI Algorithm 5.3.3 and TEKI Algorithm 5.3.4 to be discussed later) enjoys the 'initial subspace property' studied in previous works [Iglesias et al., 2013; Schillings and Stuart, 2017; Chada et al., 2020] by which, for any i and any initialization of $\{u_0^{(n)}\}_{n=1}^N$,

$$span\big(\{u_i^{(n)}\}_{n=1}^N\big) \subset span\big(\{u_0^{(n)}\}_{n=1}^N\big).$$

This can be shown easily for the IEKF Algorithm 5.3.1 by expanding the P_0^{uu} *term in* K_i *in the update formula* (5.3.6).

2. *Since we assume a Gaussian prior* $\mathcal{N}(m, P)$ *on* u*, a natural idea is to replace* $u_0^{(n)}$ *by* m *and* P_0^{uu} *by* P *in the update formula* (5.3.6)*. We pursue this idea in Section 5.4.1, where we introduce a new variant of Algorithm 5.3.1 and analyze it in the continuum limit setting. While the initial subspace property breaks down, this new variant is numerically promising, as shown in Section 5.5.*

5.3.2 Ensemble Levenberg-Marquardt optimization of data-misfit objective

Ensemble Kalman Inversion

Given an ensemble $\{u_i^{(n)}\}_{n=1}^N$, consider the following Levenberg-Marquardt update for each n:

$$u_{i+1}^{(n)} = u_i^{(n)} + v_i^{(n)}, \tag{5.3.7}$$

where $v_i^{(n)}$ is the minimizer of the following regularized (linearized) data-misfit objective (cf. equation (5.2.25))

$$\mathsf{J}_{\text{DM},i}^{(n),\text{UC}}(v) = \frac{1}{2}\big|y_i^{(n)} - h(u_i^{(n)}) - H_i v\big|_R^2 + \frac{1}{2\alpha}\big|v\big|_{P_i^{uu}}^2, \qquad y_i^{(n)} \sim \mathcal{N}(y, \alpha^{-1}R), \tag{5.3.8}$$

and $\alpha > 0$ will be regarded as a length-step. Notice that we adopt the statistical linearization (5.3.1) in the above formulation. Applying Lemma 5.2.1, we can calculate the minimizer $v_i^{(n)}$

explicitly:

$$v_i^{(n)} = (H_i^T R^{-1} H_i + \alpha^{-1}(P_i^{uu})^{-1})^{-1} H_i^T R^{-1}\big(y_i^{(n)} - h(u_i^{(n)})\big), \tag{5.3.9}$$

or, in an equivalent form,

$$v_i^{(n)} = P_i^{uu} H_i^T (H_i P_i^{uu} H_i^T + \alpha^{-1} R)^{-1}\big(y_i^{(n)} - h(u_i^{(n)})\big). \tag{5.3.10}$$

We combine (5.3.7) and (5.3.10), substitute $P_i^{uu} H_i^T = P_i^{uy}$, and make another level of approximation $H_i P_i^{uy} \approx P_i^{yy}$. This leads to the Ensemble Kalman Inversion (EKI) method [Iglesias et al., 2013].

Algorithm 5.3.3 Ensemble Kalman Inversion (EKI)

```
1: **Input**: Initial ensemble $\{u_0^{(n)}\}_{n=1}^N$, length-step $\alpha$.
2: For $i = 0, 1, \dots$ do:

        Set $K_i = P_i^{uy}(P_i^{yy} + \alpha^{-1} R)^{-1}$.
        Draw $y_i^{(n)} \sim \mathcal{N}(y, \alpha^{-1} R)$ and set
            $u_{i+1}^{(n)} = u_i^{(n)} + K_i\Big\{y_i^{(n)} - h(u_i^{(n)})\Big\}, \qquad 1 \le n \le N.$   (5.3.11)

3: **Output**: Ensemble means $m_1, m_2, \dots$
```

We note that EKI is a natural ensemble-based version of the derivative-based ILM-DM Algorithm 5.2.2. However, an important difference is that the Kalman gain in ILM-DM only uses the iterates to update H_i and P is kept fixed. In contrast, the ensemble is used in EKI to update P_i^{uy} and P_i^{yy}.

In view of the definition of K_i, we can rewrite the update (5.3.11) as a time-stepping scheme (as similarly done in [Schillings and Stuart, 2017, 2018]):

$$\begin{aligned} u_{i+1}^{(n)} &= u_i^{(n)} + \alpha P_i^{uy}(\alpha P_i^{yy} + R)^{-1}\big(y + \alpha^{-1/2}R^{1/2}\xi_i^{(n)} - h(u_i^{(n)})\big) \\ &= u_i^{(n)} + \alpha P_i^{uy}(\alpha P_i^{yy} + R)^{-1}\big(y - h(u_i^{(n)})\big) + \alpha^{1/2}P_i^{uy}(\alpha P_i^{yy} + R)^{-1}R^{1/2}\xi_i^{(n)}, \end{aligned} \tag{5.3.12}$$

where $\xi_i^{(n)} \sim \mathcal{N}(0, I)$ are independent. Taking the limit $\alpha \to 0$, we interpret (5.3.12) as a discretization of the SDE system

$$\mathrm{d}u^{(n)} = P^{uy}(t)R^{-1}\big(y - h(u^{(n)})\big)\,\mathrm{d}t + P^{uy}(t)R^{-1/2}\,\mathrm{d}W^{(n)}. \tag{5.3.13}$$

If $h(u) = Hu$ is linear, the SDE system (5.3.13) turns into

$$\mathrm{d}u^{(n)} = P^{uu}(t)H^TR^{-1}\big(y - Hu^{(n)}\big)\,\mathrm{d}t + P^{uu}(t)H^TR^{-1/2}\,\mathrm{d}W^{(n)}. \tag{5.3.14}$$

Proposition 5.3.3. *For the SDE system* (5.3.13)*, assume $h(u) = Hu$ is linear, and suppose that the initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is drawn independently from a continuous distribution with finite second moments. Then, in the large ensemble size limit $N \to \infty$ (mean-field), the distribution of $u^{(n)}(t)$ has mean $\mathfrak{m}(t)$ and covariance $\mathfrak{C}(t)$, which satisfy*

$$\frac{\mathrm{d}\mathfrak{m}(t)}{\mathrm{d}t} = \mathfrak{C}(t)H^TR^{-1}\big(y - H\mathfrak{m}(t)\big), \tag{5.3.15}$$

$$\frac{\mathrm{d}\mathfrak{C}(t)}{\mathrm{d}t} = -\mathfrak{C}(t)H^TR^{-1}H\mathfrak{C}(t). \tag{5.3.16}$$

Furthermore, the solution can be computed analytically:

$$\mathfrak{m}(t) = \Big(\mathfrak{C}(0)^{-1} + tH^T R^{-1} H\Big)^{-1} \big(\mathfrak{C}(0)^{-1}\mathfrak{m}(0) + tH^T R^{-1} y\big), \tag{5.3.17}$$

$$\mathfrak{C}(t) = \Big(\mathfrak{C}(0)^{-1} + tH^T R^{-1} H\Big)^{-1}. \tag{5.3.18}$$

In particular, if $H \in \mathbb{R}^{k\times d}$ *has full column rank (i.e.,* $d \le k$, $\operatorname{rank}(H) = d$*), then, as* $t \to \infty$,

$$\mathfrak{m}(t) \to (H^T R^{-1} H)^{-1} (H^T R^{-1} y), \tag{5.3.19}$$

$$\mathfrak{C}(t) \to 0. \tag{5.3.20}$$

If $H \in \mathbb{R}^{k\times d}$ *has full row rank (i.e.,* $d \ge k$, $\operatorname{rank}(H) = k$*), then, as* $t \to \infty$,

$$H\mathfrak{m}(t) \to y, \tag{5.3.21}$$

$$H\mathfrak{C}(t)H^T \to 0. \tag{5.3.22}$$

Proof. The proof technique is similar to [Garbuno-Inigo et al., 2020]. We will use that

$$\mathfrak{m}(t) = \lim_{N\to\infty} \mathbb{E}\big[u^{(n)}(t)\big],$$
$$\mathfrak{C}(t) = \lim_{N\to\infty} \mathbb{E}\big[e^{(n)}(t) \otimes e^{(n)}(t)\big],$$

where $e^{(n)}(t) := u^{(n)}(t) - \mathfrak{m}(t)$. First, note that (5.3.15) follows directly from (5.3.14) using that in the mean field limit $P^{uu}(t)$ can be replaced by $\mathfrak{C}(t)$. To obtain the evolution of $\mathfrak{C}(t)$, note that

$$\mathrm{d}\mathfrak{C}(t) = \lim_{N\to\infty} \mathbb{E}\Big[\mathrm{d}e^{(n)} \otimes e^{(n)} + e^{(n)} \otimes \mathrm{d}e^{(n)} + \mathrm{d}e^{(n)} \otimes \mathrm{d}e^{(n)}\Big],$$

where the last term accounts for the Itô correction. This simplifies as

$$\begin{aligned}\frac{\mathrm{d}\mathfrak{C}(t)}{\mathrm{d}t} &= \lim_{N\to\infty}\mathbb{E}\Big[-\mathfrak{C}(t)H^TR^{-1}H(e^{(n)}\otimes e^{(n)}) - (e^{(n)}\otimes e^{(n)})H^TR^{-1}H\mathfrak{C}(t)\\ &\quad + \mathfrak{C}(t)H^TR^{-1}H\mathfrak{C}(t)\Big]\\ &= -\mathfrak{C}(t)H^TR^{-1}H\mathfrak{C}(t),\end{aligned}$$

which gives Equation (5.3.16). To derive exact formulas for $\mathfrak{m}(t)$ and $\mathfrak{C}(t)$, we notice that

$$\frac{\mathrm{d}\mathfrak{C}(t)^{-1}}{\mathrm{d}t} = -\mathfrak{C}(t)^{-1}\frac{\mathrm{d}\mathfrak{C}(t)}{\mathrm{d}t}\mathfrak{C}(t)^{-1} = H^TR^{-1}H,$$

and

$$\frac{\mathrm{d}\big(\mathfrak{C}(t)^{-1}\mathfrak{m}(t)\big)}{\mathrm{d}t} = \frac{\mathrm{d}\mathfrak{C}(t)^{-1}}{\mathrm{d}t}\mathfrak{m}(t) + \mathfrak{C}(t)^{-1}\frac{\mathrm{d}\mathfrak{m}(t)}{\mathrm{d}t} = H^TR^{-1}y.$$

Then (5.3.17) and (5.3.18) follow easily.

If H has full column rank, then $H^TR^{-1}H$ is invertible and therefore, as $t\to\infty$,

$$\mathfrak{C}(t) = t^{-1}\big(t^{-1}\mathfrak{C}(0)^{-1} + H^TR^{-1}H\big)^{-1} \to 0$$

by continuity of the matrix inverse function. The limit of $\mathfrak{m}(t)$, (5.3.19), follows immediately from (5.3.17).

If H has full row rank, we make the following substitutions

$$\widetilde{\mathfrak{m}}(t) = R^{-1/2}H\mathfrak{m}(t), \qquad \widetilde{\mathfrak{C}}(t) = R^{-1/2}H\mathfrak{C}(t)H^TR^{-1/2}.$$

Then (5.3.15) and (5.3.16) can be transformed into

$$\frac{\mathrm{d}\widetilde{\mathfrak{m}}(t)}{\mathrm{d}t} = \widetilde{\mathfrak{C}}(t)\big(R^{-1/2}y - \widetilde{\mathfrak{m}}(t)\big), \tag{5.3.23}$$

$$\frac{\mathrm{d}\widetilde{\mathfrak{C}}(t)}{\mathrm{d}t} = -\widetilde{\mathfrak{C}}(t)^2. \tag{5.3.24}$$

Using the fact that $\widetilde{\mathfrak{C}}(0) = R^{-1/2}H\mathfrak{C}(0)H^T R^{-1/2}$ is invertible, we can solve these using the same technique as in the previous case:

$$\widetilde{\mathfrak{C}}(t) = \big(\widetilde{\mathfrak{C}}(0)^{-1} + t\big)^{-1}, \qquad \widetilde{\mathfrak{m}}(t) = \big(\widetilde{\mathfrak{C}}(0)^{-1} + t\big)^{-1}\big(\widetilde{\mathfrak{C}}(0)^{-1}\widetilde{\mathfrak{m}}(0) + tR^{-1/2}y\big).$$

As $t \to \infty$, we have $\widetilde{\mathfrak{m}}(t) \to R^{-1/2}y$ and $\widetilde{\mathfrak{C}}(t) \to 0$, which lead to (5.3.21) and (5.3.22). $\square$

Remark 5.3.4. *Some remarks:*

1. *In fact,* (5.3.22) *always holds, without rank constraints on H. The proof follows the similar idea as in [Schillings and Stuart, 2017]. This can be viewed from Equation* (5.3.24)*, where we can perform eigenvalue decomposition $\widetilde{\mathfrak{C}}(0) = X\Lambda(0)X^T$, and show that $\widetilde{\mathfrak{C}}(t) = X\Lambda(t)X^T$, where $\Lambda(t)$ are diagonal matrices and $\Lambda(t) \to 0$ as $t \to \infty$. The statement that $H\mathfrak{C}(t)H^T \to 0$ (or $\mathfrak{C}(t) \to 0$ if H has full column rank) is referred to as 'ensemble collapse'. This can be interpreted as 'the images of all the particles under H collapse to a single point as time evolves'. Our numerical results in Section 5.5 show that the ensemble collapse phenomenon of Algorithm 5.3.3 is also observed in a variety of nonlinear examples and outside the mean-field limit, with moderate ensemble size N. These empirical results justify the practical significance of the linear continuum analysis in Proposition 5.3.3.*

2. *Under the same setting of Proposition 5.3.3, if we further require that the initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is drawn independently from the prior distribution $\mathcal{N}(m, P)$, then in*

the mean-field limit $N \to \infty$ we have $\mathfrak{m}(0) = m$ and $\mathfrak{C}(0) = P$, leading to

$$\begin{aligned} \mathfrak{m}(1) &= (P^{-1} + H^T R^{-1} H)^{-1}(P^{-1}m + H^T R^{-1} y), \\ \mathfrak{C}(1) &= (P^{-1} + H^T R^{-1} H)^{-1} \end{aligned}$$

which are the true posterior mean and covariance, respectively. However, we have observed in a variety of numerical examples (not reported here) that in nonlinear problems it is often necessary to run EKI up to times larger than 1 to obtain adequate approximation of the posterior mean and covariance. Providing a suitable stopping criteria for EKI is a topic of current research [Schillings and Stuart, 2018; Iglesias and Yang, 2021] beyond the scope of our work.

5.3.3 Ensemble Levenberg-Marquardt optimization of Tikhonov-Phillips objective

Tikhonov Ensemble Kalman Inversion

Recall that we define

$$z := \begin{bmatrix} y \\ m \end{bmatrix}, \qquad g(u) := \begin{bmatrix} h(u) \\ u \end{bmatrix}, \qquad Q := \begin{bmatrix} R & 0 \\ 0 & P \end{bmatrix}.$$

Then, given an ensemble $\{u_i^{(n)}\}_{n=1}^N$, we can define

$$g_i = \frac{1}{N} \sum_{n=1}^{N} g(u_i^{(n)}),$$

and empirical covariances

$$
\begin{aligned}
P_i^{zz} &= \frac{1}{N}\sum_{n=1}^{N}\big(g(u_i^{(n)}) - g_i\big)\big(g(u_i^{(n)}) - g_i\big)^T, \\
P_i^{uz} &= \frac{1}{N}\sum_{n=1}^{N}\big(u_i^{(n)} - m_i\big)\big(g(u_i^{(n)}) - g_i\big)^T.
\end{aligned}
$$

Furthermore, we define the statistical linearization G_i:

$$
g'(u_i^{(n)}) \approx (P_i^{uz})^T (P_i^{uu})^{-1} =: G_i. \tag{5.3.25}
$$

It is not hard to check that

$$
G_i = \begin{bmatrix} H_i \\ I \end{bmatrix},
$$

with H_i defined in (5.3.1).

Given an ensemble $\{u_i^{(n)}\}_{n=1}^N$, consider the following Levenberg-Marquardt update for each n:

$$
u_{i+1}^{(n)} = u_i^{(n)} + v_i^{(n)},
$$

where $v_i^{(n)}$ is the minimizer of the following regularized (linearized) Tikhonov-Phillips objective (cf. equation (5.2.29))

$$
\mathsf{J}_{\text{TP},i}^{(n),\text{UC}}(v) = \frac{1}{2}\big|z_i^{(n)} - g(u_i^{(n)}) - G_i v\big|_Q^2 + \frac{1}{2\alpha}|v|_{P_i^{uu}}^2, \qquad z_i^{(n)} \sim \mathcal{N}(z, \alpha^{-1}Q), \tag{5.3.26}
$$

and $\alpha > 0$ will be regarded as a length-step. We can calculate the minimizer $v_i^{(n)}$ explicitly, applying Lemma 5.2.1:

$$
v_i^{(n)} = (G_i^T Q^{-1} G_i + \alpha^{-1}(P_i^{uu})^{-1})^{-1} G_i^T Q^{-1}\big(z_i^{(n)} - g(u_i^{(n)})\big), \tag{5.3.27}
$$

or, in an equivalent form,

$$v_i^{(n)} = P_i^{uu} G_i^T (G_i P_i^{uu} G_i^T + \alpha^{-1} Q)^{-1} \big(z_i^{(n)} - g(u_i^{(n)})\big). \tag{5.3.28}$$

Similar to EKI, in Equation (5.3.28) we substitute $P_i^{uu} G_i^T = P_i^{uz}$, and make the approximation $G_i P_i^{uz} \approx P_i^{zz}$. This leads to Tikhonov Ensemble Kalman Inversion (TEKI), described in Algorithm 5.3.4.

Algorithm 5.3.4 Tikhonov Ensemble Kalman Inversion (TEKI)

```
1: **Input**: Initial ensemble $\{u_0^{(n)}\}_{n=1}^N$, length-step $\alpha$.
2: For $i = 0, 1, \dots$ do:
       Set $K_i = P_i^{uz}(P_i^{zz} + \alpha^{-1} Q)^{-1}$.
       Draw $z_i^{(n)} \sim \mathcal{N}(z, \alpha^{-1} Q)$ and set
           $u_{i+1}^{(n)} = u_i^{(n)} + K_i \Big\{ z_i^{(n)} - g(u_i^{(n)}) \Big\}, \qquad 1 \le n \le N.$   (5.3.29)
3: **Output**: Ensemble means $m_1, m_2, \dots$
```

We note that TEKI is a natural ensemble-based version of the derivative-based ILM-TP Algorithm 5.2.3. However, the Kalman gain in ILM-TP keeps P fixed, while in TEKI P_i^{uz} and P_i^{zz} are updated using the ensemble.

Analysis of TEKI

When α is small, we can rewrite the update (5.3.29) as a time-stepping scheme (as similarly done in [Chada et al., 2020]):

$$\begin{aligned} u_{i+1}^{(n)} &= u_i^{(n)} + \alpha P_i^{uz} (\alpha P_i^{zz} + Q)^{-1} \big(z + \alpha^{-1/2} Q^{1/2} \xi_i^{(n)} - g(u_i^{(n)})\big) \\ &= u_i^{(n)} + \alpha P_i^{uz} (\alpha P_i^{zz} + Q)^{-1} \big(z - g(u_i^{(n)})\big) + \alpha^{1/2} P_i^{uz} (\alpha P_i^{zz} + Q)^{-1} Q^{1/2} \xi_i^{(n)}, \end{aligned} \tag{5.3.30}$$

where $\xi_i^{(n)} \sim \mathcal{N}(0, I_{d+k})$ are independent. Taking the limit $\alpha \to 0$, we can interpret (5.3.30) as a discretization of an interacting particle SDE system

$$\mathrm{d}u^{(n)} = P^{uz}(t)Q^{-1}\big(z - g(u^{(n)})\big)\,\mathrm{d}t + P^{uz}(t)Q^{-1/2}\,\mathrm{d}W^{(n)}. \tag{5.3.31}$$

If $h(u) = Hu$ is linear, $g(u) = Gu$ is also linear and the SDE system (5.3.31) can be rewritten as

$$\begin{aligned}\mathrm{d}u^{(n)} &= P^{uz}(t)Q^{-1}\big(z - Gu^{(n)}\big)\,\mathrm{d}t + P^{uz}(t)Q^{-1/2}\,\mathrm{d}W^{(n)}\\ &= P^{uu}(t)G^TQ^{-1}\big(z - Gu^{(n)}\big)\,\mathrm{d}t + P^{uu}(t)G^TQ^{-1/2}\,\mathrm{d}W^{(n)}.\end{aligned} \tag{5.3.32}$$

Proposition 5.3.5. *For the SDE system* (5.3.31)*, assume $h(u) = Hu$ is linear, and that the initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is made of independent samples from a distribution with finite second moments. Then, in the large ensemble limit (mean-field), the distribution of $u^{(n)}(t)$ has mean $\mathfrak{m}(t)$ and covariance $\mathfrak{C}(t)$, which satisfy:*

$$\frac{\mathrm{d}\mathfrak{m}(t)}{\mathrm{d}t} = \mathfrak{C}(t)G^TQ^{-1}(z - G\mathfrak{m}(t)), \tag{5.3.33}$$

$$\frac{\mathrm{d}\mathfrak{C}(t)}{\mathrm{d}t} = -\mathfrak{C}(t)G^TQ^{-1}G\mathfrak{C}(t). \tag{5.3.34}$$

Furthermore, the solution can be computed analytically:

$$\mathfrak{m}(t) = \Big(\mathfrak{C}(0)^{-1} + tG^TQ^{-1}G\Big)^{-1}\big(\mathfrak{C}(0)^{-1}\mathfrak{m}(0) + tG^TQ^{-1}z\big), \tag{5.3.35}$$

$$\mathfrak{C}(t) = \Big(\mathfrak{C}(0)^{-1} + tG^TQ^{-1}G\Big)^{-1}. \tag{5.3.36}$$

In particular, as $t \to \infty$,

$$\begin{aligned}\mathfrak{m}(t) &\to (G^TQ^{-1}G)^{-1}(G^TQ^{-1}z)\\ &= (H^TR^{-1}H + P)^{-1}(H^TR^{-1}y + P^{-1}m),\end{aligned} \tag{5.3.37}$$

$$\mathfrak{C}(t) \to 0. \tag{5.3.38}$$

Notice that $\mathfrak{m}(t)$ converges to the true posterior mean.

Proof. Equations (5.3.33)-(5.3.36) can be derived similarly as in the proof of Proposition 5.3.3, replacing H by G, R by Q, and y by z. Now since $G^TQ^{-1}G = H^TR^{-1}H + P$ is always invertible we have that, as $t \to \infty$,

$$\mathfrak{C}(t) = t^{-1}\big(t^{-1}\mathfrak{C}(0)^{-1} + G^TQ^{-1}G\big)^{-1} \to 0$$

by continuity of the matrix inverse function. The limit of $\mathfrak{m}(t)$ in (5.3.37) follows directly from (5.3.35). □

5.4 Ensemble Kalman methods: New variants

In the previous section we discussed three popular subfamilies of iterative ensemble Kalman methods, analogous to the derivative-based algorithms in Section 5.2. The aim of this section is to introduce two new iterative ensemble Kalman methods which are inspired by the SDE continuum limit structure of the algorithms in Section 5.3. The two new methods that we introduce have in common that they rely on statistical linearization, and that the long-time limit of the ensemble covariance recovers the posterior covariance in a linear setting. This holds true even if the initial ensemble is not drawn from the prior distribution on the unknown.

Subsection 5.4.1 contains a new variant of IEKF which in addition to recovering the posterior covariance, it also recovers the posterior mean in the long-time limit. Subsection 5.4.2 introduces a new variant of the EKI method. Finally, Subsection 5.4.3 highlights the gradient structure of the algorithms in this and the previous section, shows that our new variant of IEKF can also be interpreted as a modified TEKI algorithm, and sets our proposed new methods into the broader literature.

5.4.1 Iterative Ensemble Kalman Filter with statistical linearization

In some of the literature on iterative ensemble Kalman methods [Reynolds et al., 2006; Ungarala, 2012], the length-step (α in Algorithms 5.3.1 and 5.3.2) is set to be 1. Although this choice of length-step allows to recover the true posterior mean in the linear case after one iteration (see Section 5.3.1), it leads to numerical instability in complex nonlinear models. Alternative ways to set α include performing a line-search that satisfies Wolfe's condition, or using other ad-hoc line-search criteria [Gu and Oliver, 2007]. These methods allow α to be adaptively chosen throughout the iterations, but they introduce other hyperparameters that need to be selected manually.

Our idea here is to slightly modify Algorithm 5.3.1 so that in the linear case its continuum limit has the true posterior as its invariant distribution. In this way we can simply choose a small enough α and run the algorithm until the iterates reach a statistical equilibrium, avoiding the need to specify suitable hyperparameters and stopping criteria. Our empirical results show that this approach also performs well in the nonlinear case.

In the update Equation (5.3.6), we replace each of the $u_0^{(n)}$ by a perturbation of the prior mean m, and we replace P_0^{uu} by the prior covariance matrix P in the definition of K_i. Details can be found below in our modified algorithm, which we call IEKF-SL.

It is natural to regard the update (5.4.1) as a time-stepping scheme. We rewrite it in an

Algorithm 5.4.1 Iterative Ensemble Kalman Filter with Statistical Linearization (IEKF-SL)

```
1: Input: Initial ensemble $\{u_0^{(n)}\}_{n=1}^N$, step size $\alpha$.
2: For $i = 0, 1, \dots$ do:
       Set $K_i = PH_i^T(H_iPH_i^T + R)^{-1}, \qquad H_i = (P_i^{uy})^T(P_i^{uu})^{-1}.$
       Draw $y_i^{(n)} \sim \mathcal{N}(y, 2\alpha^{-1}R)$, $m_i^{(n)} \sim \mathcal{N}(m, 2\alpha^{-1}P)$ and set
       $u_{i+1}^{(n)} = u_i^{(n)} + \alpha\Big\{K_i\big(y_i^{(n)} - h(u_i^{(n)})\big) + (I - K_iH_i)\big(m_i^{(n)} - u_i^{(n)}\big)\Big\}, \qquad 1 \le n \le N. \quad (5.4.1)$
3: Output: Ensemble means $m_1, m_2, \dots$
```

alternative form, in analogy to (5.3.4):

$$
\begin{aligned}
u_{i+1}^{(n)} &= u_i^{(n)} + \alpha C_i\Big\{H_i^TR^{-1}\big(y_i^{(n)} - h(u_i^{(n)})\big) + P^{-1}\big(m_i^{(n)} - u_i^{(n)}\big)\Big\}\\
&= u_i^{(n)} + \alpha C_i\Big\{H_i^TR^{-1}\big(y - h(u_i^{(n)})\big) + P^{-1}\big(m - u_i^{(n)}\big) + (H_i^TR^{-1}\zeta + P^{-1}\eta)\Big\},
\end{aligned}
\tag{5.4.2}
$$

where

$$
C_i = \big(H_i^TR^{-1}H_i + P^{-1}\big)^{-1}, \quad \zeta \sim \mathcal{N}(0, 2\alpha^{-1}R), \quad \eta \sim \mathcal{N}(0, 2\alpha^{-1}P).
$$

We interpret Equation (5.4.2) as a discretization of the SDE system

$$
\mathrm{d}u^{(n)} = C(t)\Big(H(t)^TR^{-1}\big(y - h(u^{(n)})\big) + P^{-1}\big(m - u^{(n)}\big)\Big)\,\mathrm{d}t + \sqrt{2C(t)}\,\mathrm{d}W^{(n)}, \tag{5.4.3}
$$

where

$$
\begin{aligned}
H(t) &= \big(P^{uy}(t)\big)^T\big(P^{uu}(t)\big)^{-1},\\
C(t) &= \big(H(t)^TR^{-1}H(t) + P^{-1}\big)^{-1}.
\end{aligned}
$$

The diffusion term can be derived using the fact that

$$C_i(H_i^T R^{-1}\zeta + P^{-1}\eta) \sim \mathcal{N}\big(0, 2\alpha^{-1}C_i(H_i^T R^{-1}H_i + P^{-1})C_i\big) = \mathcal{N}(0, 2\alpha^{-1}C_i).$$

If $h(u) = Hu$ is linear and the empirical covariance $P^{uu}(t)$ has full rank for all t, then $H(t) \equiv H$, $C(t) \equiv C = (P^{-1} + H^T R^{-1} H)^{-1}$, and the SDE system can be decoupled and further simplified:

$$\begin{aligned} \mathrm{d}u^{(n)} &= C\Big(H^T R^{-1}\big(y - Hu^{(n)}\big) + P^{-1}(m - u^{(n)})\Big)\,\mathrm{d}t + \sqrt{2C}\,\mathrm{d}W \\ &= \Big(-u^{(n)} + C(H^T R^{-1}y + P^{-1}m)\Big)\,\mathrm{d}t + \sqrt{2C}\,\mathrm{d}W. \end{aligned} \tag{5.4.4}$$

Proposition 5.4.1. *For the SDE system* (5.4.3), *assume $h(u) = Hu$ is linear and $H(t) \equiv H$ holds. Assume that the initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is made of independent samples from a continuous distribution with finite second moments. Then, for $1 \le n \le N$, the mean $\mathfrak{m}(t)$ and covariance $\mathfrak{C}(t)$ of $u^{(n)}(t)$ satisfy*

$$\frac{\mathrm{d}\mathfrak{m}(t)}{\mathrm{d}t} = -\mathfrak{m}(t) + C(H^T R^{-1}y + P^{-1}m), \tag{5.4.5}$$

$$\frac{\mathrm{d}\mathfrak{C}(t)}{\mathrm{d}t} = -2\mathfrak{C}(t) + 2C. \tag{5.4.6}$$

Furthermore, as $t \to \infty$,

$$\begin{aligned} \mathfrak{m}(t) &\to C(H^T R^{-1}y + P^{-1}m) \\ &= (P^{-1} + H^T R^{-1}H)^{-1}(H^T R^{-1}y + P^{-1}m), \end{aligned} \tag{5.4.7}$$

$$\mathfrak{C}(t) \to C = (P^{-1} + H^T R^{-1}H)^{-1}. \tag{5.4.8}$$

In other words, $\mathfrak{m}(t)$ and $\mathfrak{C}(t)$ converge to the true posterior mean and covariance, respectively.

Proof. It is clear that, for fixed $t > 0$, $\{u^{(n)}(t)\}_{n=1}^N$ are independent and identically distributed. The evolution of the mean follows directly from (5.4.4), and the evolution of the covariance follows from (5.4.4) by applying Itô's formula. It is then straightforward to derive (5.4.7) and (5.4.8) from (5.4.5) and (5.4.6), respectively. □

5.4.2 Ensemble Kalman inversion with statistical linearization

Recall that in the formulation of EKI, we define a regularized (linearized) data-misfit objective (5.3.8), where we have a regularizer on v with respect to the norm $|\cdot|_{P_i^{uu}}$. However, in view of Proposition 5.3.3, under a linear forward model $h(u) = Hu$, the particles $\{u_i^{(n)}\}_{i=1}^n$ will 'collapse', meaning that the empirical covariance of $\{Hu_i^{(n)}\}_{n=1}^N$ will vanish in the large i limit. One possible solution to this is 'covariance inflation', namely to inject certain amount of random noise after each ensemble update. However, this requires ad-hoc tuning of additional hyperparameters. An alternative approach to avoid the ensemble collapse is to modify the regularization term in the Levenberg-Marquardt formulation (5.3.8). The rough idea is to consider another regularizer on $H_i v$ in the data space, as we describe in what follows.

We define a new regularized data-misfit objective, slightly different from (5.3.8):

$$\mathsf{J}_{\text{DM},i}^{(n),\text{UC}}(v) = \frac{1}{2}|y_i^{(n)} - h(u_i^{(n)}) - H_i v|_R^2 + \frac{1}{2\alpha}|v|_{C_i}^2 \qquad y_i^{(n)} \sim \mathcal{N}(y, 2\alpha^{-1}R), \tag{5.4.9}$$

where

$$C_i = (P^{-1} + H_i^T R^{-1} H_i)^{-1}, \tag{5.4.10}$$

and H_i is defined in (5.3.1). The regularization term can be decomposed as

$$|v|_{C_i}^2 = |v|_P^2 + |H_i v|_R^2.$$

The first term can be regarded as a regularization on v with respect to the prior covariance

P. The second term can be regarded as a regularization on $H_i v$ with respect to the noise covariance R. Applying Lemma 5.2.1, we can calculate the minimizer of (5.4.9):

$$\begin{aligned} v_i^{(n)} &= (H_i^T R^{-1} H_i + \alpha^{-1} C_i^{-1})^{-1} H_i^T R^{-1} \big(y_i^{(n)} - h(u_i^{(n)})\big) \\ &= \big(\alpha^{-1} P^{-1} + (1+\alpha^{-1}) H_i^T R^{-1} H_i\big)^{-1} H_i^T R^{-1} \big(y_i^{(n)} - h(u_i^{(n)})\big) \\ &= \alpha P H_i^T \big((1+\alpha) H_i P H_i^T + R\big)^{-1} \big(y_i^{(n)} - h(u_i^{(n)})\big), \end{aligned} \tag{5.4.11}$$

where the second equality follows from the definition of C_i, and the third equality follows from the matrix identity (5.2.7). This leads to Algorithm 5.4.2.

Algorithm 5.4.2 Ensemble Kalman Inversion with Statistical Linearization (EKI-SL)

```
1: **Input**: Initial ensemble $\{u_0^{(n)}\}_{n=1}^N$, step size $\alpha$.
2: For $i = 0, 1, \dots$ do:
       Set $K_i = \alpha P H_i^T \big((1+\alpha) H_i P H_i^T + R\big)^{-1}$,    $H_i = (P_i^{uy})^T (P_i^{uu})^{-1}$.
       Draw $y_i^{(n)} \sim \mathcal{N}(y, 2\alpha^{-1} R)$ and set
           $u_{i+1}^{(n)} = u_i^{(n)} + K_i \Big\{ y_i^{(n)} - h(u_i^{(n)}) \Big\}, \quad 1 \le n \le N.$   (5.4.12)
3: **Output**: Ensemble means $m_1, m_2, \dots$
```

For small $\alpha > 0$, we interpret the update (5.4.12) as a discretization of the coupled SDE system

$$\begin{aligned} \mathrm{d}u^{(n)} &= P H(t)^T \big(H(t) P H(t)^T + R\big)^{-1} \Big(\big(y - h(u^{(n)})\big)\,\mathrm{d}t + \sqrt{2} R^{1/2}\,\mathrm{d}W \Big) \\ &= C(t) H(t)^T R^{-1} \big(y - h(u^{(n)})\big)\,\mathrm{d}t + \sqrt{2} C(t) H(t)^T R^{-1/2}\,\mathrm{d}W, \end{aligned} \tag{5.4.13}$$

where

$$
\begin{aligned}
H(t) &= \big(P^{uy}(t)\big)^T \big(P^{uu}(t)\big)^{-1},\\
C(t) &= \big(P^{-1} + H(t)^T R^{-1} H(t)\big)^{-1}.
\end{aligned}
$$

For our next result we will work under the assumption that $H(t) \equiv H$ is constant, which in particular requires that the empirical covariance $P^{uu}(t)$ has full rank for all t. Importantly, under this assumption $C(t) \equiv C = (P^{-1} + H^T R^{-1} H)^{-1}$and the SDE system is decoupled:

$$
\mathrm{d}u^{(n)} = CH^T R^{-1}\big(y - Hu^{(n)}\big)\,\mathrm{d}t + \sqrt{2}CH^T R^{-1/2}\,\mathrm{d}W^{(n)}. \tag{5.4.14}
$$

Proposition 5.4.2. *For the SDE system* (5.4.13)*, assume $h(u) = Hu$ is linear and $H(t) \equiv H$ holds. Assume that the initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is made of independent samples from a continuous distribution with finite second moments. Then the mean $\mathfrak{m}(t)$ and covariance $\mathfrak{C}(t)$ of $u^{(n)}(t)$ satisfy*

$$
\frac{\mathrm{d}\mathfrak{m}(t)}{\mathrm{d}t} = CH^T R^{-1}(y - H\mathfrak{m}(t)), \tag{5.4.15}
$$

$$
\frac{\mathrm{d}\mathfrak{C}(t)}{\mathrm{d}t} = -CH^T R^{-1} H\mathfrak{C}(t) - \mathfrak{C}(t)H^T R^{-1} HC + 2CH^T R^{-1} HC. \tag{5.4.16}
$$

In particular, if $H \in \mathbb{R}^{k\times d}$ has full column rank (i.e., $d \le k$, $\mathrm{rank}(H) = d$), then, as $t \to \infty$,

$$
\mathfrak{m}(t) \to (H^T R^{-1} H)^{-1}(H^T R^{-1} y), \tag{5.4.17}
$$

$$
\mathfrak{C}(t) \to C = (P^{-1} + H^T R^{-1} H)^{-1}. \tag{5.4.18}
$$

If $H \in \mathbb{R}^{k\times d}$ *has full row rank (i.e.,* $d \geq k$, $\operatorname{rank}(H) = k$*), then, as* $t \to \infty$,

$$H\mathfrak{m}(t) \to y, \tag{5.4.19}$$

$$H\mathfrak{C}(t)H^T \to HCH^T. \tag{5.4.20}$$

Proof. Note that here, in contrast to the setting of Proposition 5.3.3, the distribution of $u^{(n)}(t)$ does not depend on the ensemble size N, and we have simply $\mathfrak{m}(t) = \mathbb{E}\big[u^{(n)}(t)\big]$ and $\mathfrak{C}(t) = \mathbb{E}\left[e^{(n)}(t) \otimes e^{(n)}(t)\right]$, with $e^{(n)}(t) := u^{(n)}(t) - \mathfrak{m}(t)$. The evolution of $\mathfrak{m}(t)$ follows directly from (5.4.14). To obtain the evolution of $\mathfrak{C}(t)$, we use a similar technique as in the proof of Proposition 5.3.3. Applying Itô's formula,

$$\begin{aligned}\frac{\mathrm{d}\mathfrak{C}(t)}{\mathrm{d}t} &= \mathbb{E}\Big[-CH^TR^{-1}H(e^{(n)} \otimes e^{(n)}) - (e^{(n)} \otimes e^{(n)})H^TR^{-1}HC + 2CH^TR^{-1}HC\Big]\\ &= -CH^TR^{-1}H\mathfrak{C}(t) - \mathfrak{C}(t)H^TR^{-1}HC + 2CH^TR^{-1}HC,\end{aligned}$$

which recovers (5.4.16).

If H has full column rank, then $H^TR^{-1}H$ is invertible, and (5.4.17) follows immediately from (5.4.15). By setting the right-hand side of (5.4.16) to 0, and using the fact that it has a unique solution, we derive (5.4.18).

If H has full row rank, then (5.4.19) follows immediately from (5.4.15). Next, the substitutions

$$\widetilde{\mathfrak{C}}(t) = R^{-1/2}H\mathfrak{C}(t)H^TR^{-1/2}, \qquad \widetilde{C} = R^{-1/2}HCH^TR^{-1/2}$$

allow to transform (5.4.16) into

$$\frac{\mathrm{d}\widetilde{\mathfrak{C}}(t)}{\mathrm{d}t} = -\widetilde{C}\widetilde{\mathfrak{C}}(t) - \widetilde{\mathfrak{C}}(t)\widetilde{C} + 2\widetilde{C}^2. \tag{5.4.21}$$

Since $\widetilde{C}$ is invertible, by setting the right-hand side of (5.4.21) to 0 we deduce that $\widetilde{\mathfrak{C}}(t) \to \widetilde{C}$ as $t \to \infty$, which recovers (5.4.20). □

5.4.3 Gradient structure and discussion

LM Algorithms in the Continuum Limit Levenberg-Marquardt algorithms have a natural gradient structure in the continuum limit. This was shown in Equation (5.2.27), where the preconditioner P corresponds to the regularizer $|\cdot|_P$ that is used in the Levenberg-Marquardt algorithm (5.2.22). Ensemble-based Levenberg-Marquardt algorithms also possess a gradient structure. To see this, we consider an update $u_{i+1}^{(n)} = u_i^{(n)} + v_i^{(n)}$, where $v_i^{(n)}$ is the unconstrained minimizer of the same objective as (5.4.9)

$$\mathsf{J}_{\mathrm{DM},i}^{(n),\mathrm{UC}}(v) = \frac{1}{2}\big|y_i^{(n)} - h(u_i^{(n)}) - H_i v\big|_R^2 + \frac{1}{2\alpha}|v|_{S_i}^2 \qquad y_i^{(n)} \sim \mathcal{N}(y, 2\alpha^{-1}R),$$

except that we allow any positive (semi)definite matrix S_i to act as a 'regularizer'. The resulting continuum limit is given by the SDE system

$$\mathrm{d}u^{(n)} = S(t)H(t)^T R^{-1}\big(y - h(u^{(n)})\big)\,\mathrm{d}t + \sqrt{2}S(t)H(t)^T R^{-1/2}\,\mathrm{d}W^{(n)}, \tag{5.4.22}$$

where $u^{(n)}(t), S(t), H(t)$ are continuous time analogs of $u_i^{(n)}, S_i, H_i$.

Notice that $H(t)^T R^{-1}(y-h(u^{(n)}))$ is exactly $-\mathsf{J}'_{\mathrm{DM}}(u^{(n)})$, so that we may rewrite (5.4.22) as

$$\dot{u}^{(n)} = -S(t)\mathsf{J}'_{\mathrm{DM}}(u^{(n)}) + \sqrt{2}S(t)H(t)^T R^{-1/2}\dot{W}^{(n)}, \tag{5.4.23}$$

which is a perturbed gradient descent with preconditioner $S(t)$. We recall that $S(t) = P^{uu}(t)$ in EKI and $S(t) = C(t) = \big(P^{-1} + H(t)^T R^{-1} H(t)\big)^{-1}$ in EKI-SL. Other choices of $S(t)$ are possible and will be studied in future work.

Gauss-Newton Algorithms in the Continuum Limit Gauss-Newton algorithms also have a natural gradient structure in the continuum limit. As we have shown in Equation (5.2.21), the Gauss-Newton method applied to a Tikhonov-Phillips objective can be regarded as a gradient flow with a preconditioner that is the inverse Hessian matrix of the objective. Ensemble-based Gauss-Newton methods also possess a similar gradient structure. Recall that in Equation (5.4.3) we formulate the continuum limit of IEKF-SL:

$$\mathrm{d}u^{(n)} = C(t)\Big(H(t)^T R^{-1}\big(y - h(u^{(n)})\big) + P^{-1}\big(m - u^{(n)}\big)\Big)\,\mathrm{d}t + \sqrt{2C(t)}\,\mathrm{d}W^{(n)}. \quad (5.4.24)$$

Notice that the drift term is exactly $-C(t)\mathsf{J}'_{\text{TP}}(u^{(n)})$, so we may rewrite (5.4.24) as

$$\dot{u}^{(n)} = -C(t)\mathsf{J}'_{\text{TP}}(u^{(n)}) + \sqrt{2C(t)}\dot{W}^{(n)}, \quad (5.4.25)$$

which is a perturbed gradient descent with preconditioner $C(t)$, the inverse Hessian of J_{TP}.

A Unified View of Levenberg-Marquardt and Gauss-Newton Algorithms in the Continuum Limit As a conclusion of above discussion, in the continuum limit Levenberg-Marquardt algorithms (e.g., EKI, EKI-SL) are (perturbed) gradient descent methods, with a preconditioner determined by the regularizer used in the Levenberg-Marquardt update step. Gauss-Newton algorithms (e.g., IEKF, IEKF-SL) are also (perturbed) gradient descent, with a preconditioner determined by the inverse Hessian of the objective.

Interestingly, there are cases when the two types of algorithms coincide, even with the same amount of perturbation in the gradient descent step. A way to see this is to set the Levenberg-Marquardt regularizer to be the inverse Hessian of the objective. In equation (5.4.23), if we consider the Tikhonov-Phillips objective (i.e., replace J_{DM} by J_{TP}, H by G, Q by R), and set $S(t) = C(t)$, then (5.4.23) and (5.4.25) will coincide, leading to the same SDE system. This is the reason why we do not introduce a 'TEKI-SL' algorithm, as it is identical to IEKF-SL.

Relationship to Ensemble Kalman Sampler (EKS) Another algorithm of interest is the Ensemble Kalman Sampler (EKS) [Garbuno-Inigo et al., 2020; Ding and Li, 2021]. Although not discussed in this work, the EKS update is similar to (5.4.24). In fact, if we replace $C(t)$ by the ensemble covariance $P^{uu}(t)$, and use the fact that $P^{uu}(t)H(t)^T = P^{uy}(t)$ by definition, we immediately recover the evolution equation of EKS:

$$\mathrm{d}u^{(n)} = \Big(P^{uy}(t)^T R^{-1}\big(y - h(u^{(n)})\big) + P^{-1}\big(m - u^{(n)}\big)\Big)\,\mathrm{d}t + \sqrt{2P^{uu}(t)}\,\mathrm{d}W^{(n)}.$$

Then an Euler-Maruyama discretization is performed to compute the update formula in discrete form. However, as we find out in several numerical experiments, the length-step α needs to be chosen carefully. If the noise R has a small scale, EKS often blows up in the first few iterations, while other algorithms with the same length-step do not. Also, if we consider a linear forward model $h(u) = Hu$, EKS still requires a mean field assumption $N \to \infty$ in order for it to converge to the posterior distribution, while IEKF-SL approximately needs $N \approx d$ particles where d is the input dimension.

5.5 Numerical examples

In this section we provide a numerical comparison of the iterative ensemble Kalman methods introduced in Sections 5.3 and 5.4. Our experiments highlight the variety of applications that have motivated the development of these methods, and illustrate their use in a wide range of settings. In order to provide a comparison between all variants, we will assess the performance in various ways. First, the relative error at artificial time $t > 0$ is defined as

$$\text{relative error} = \frac{|m(t) - u^\dagger|}{|u^\dagger|},$$

where $m(t)$ denotes the mean of the ensemble at time t. Plots of the evolution of the data misfit $\mathsf{J}_{\text{DM}}\big(m(t)\big)$ and Tikhonov-Phillips $\mathsf{J}_{\text{TP}}\big(m(t)\big)$ objectives will also be provided. We will also assess the performance through reconstruction plots, which show the ensemble mean and several ensemble quantiles at the last iteration. Additionally, for the first experiment we compare each variant through the evolution of the Frobenius norm $\|P^{uu}(t)\|_F$ of the ensemble covariance.

5.5.1 Elliptic boundary value problem

In this subsection we consider a simple nonlinear Bayesian inverse problem originally presented in [Ernst et al., 2015], where the unknown parameter is two dimensional and the forward map admits a closed formula. These features facilitate both the visualization and the interpretation of the solution.

Problem setup

Consider the elliptic boundary value problem

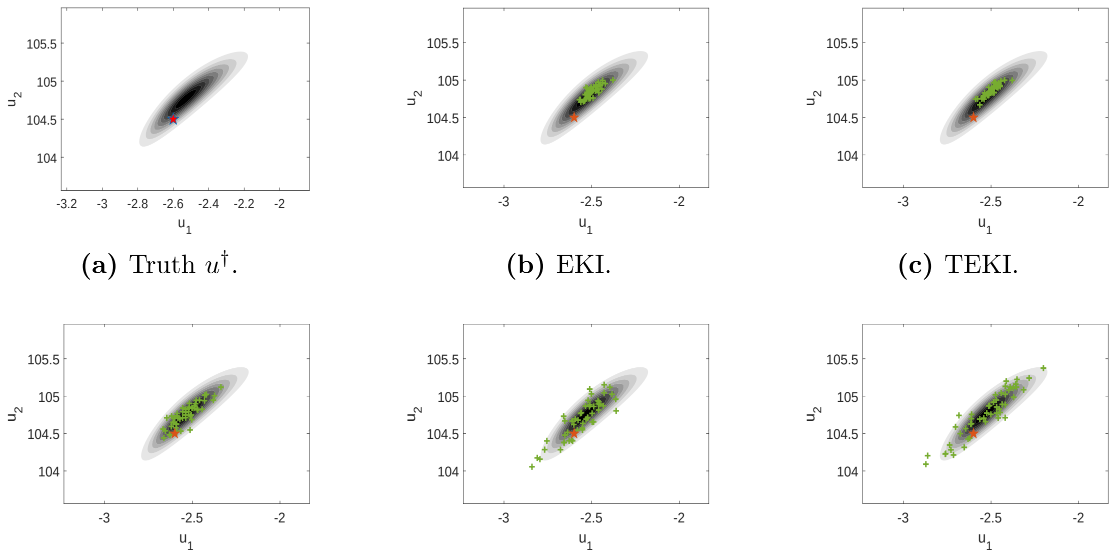
$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x}\left(\exp(u_1)\frac{\mathrm{d}}{\mathrm{d}x}p(x)\right) &= 1, \quad x \in (0,1), \\ p(0) = 0, \qquad p(1) &= u_2. \end{aligned} \tag{5.5.1}$$

It can be checked that (5.5.1) has an explicit solution

$$p_u(x) = u_2 x - \frac{1}{2}\exp(-u_1)(x^2 - x). \tag{5.5.2}$$

We seek to recover $u = (u_1, u_2)^T$ from noisy observation of p_u at two points $x_1 = 0.25$ and $x_2 = 0.75$. Precisely, we define $h(u) := \big(p_u(x_1), p_u(x_2)\big)^T$ and consider the inverse problem of recovering u from data y of the form

$$y = h(u) + \eta, \tag{5.5.3}$$



(a) Truth $u^\dagger$. **(b)** EKI. **(c)** TEKI. **(d)** IEKF. **(e)** EKI-SL. **(f)** IEKF-SL.

Figure 5.1: Ensemble members (green) after 100 iterations, with truth $u^\dagger$ (red star) and contour plot of (unnormalized) posterior density.

where $\eta \sim \mathcal{N}(0, \gamma^2 I_2)$. We set a Gaussian prior on the unknown parameter $u \sim \mathcal{N}(0,1) \times \mathcal{N}(100, 16)$. In our numerical experiments we let the true parameter be $u^\dagger = (-2.6, 104.5)^T$ and use it to generate synthetic data $y = h(u^\dagger) + \eta$ with noise level $\gamma = 0.1$.

Implementation details and numerical results

We set the ensemble size to be $N = 50$. The initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is drawn independently from the prior. The length-step α is fixed to be 0.1 for all methods. We run each algorithm up to time $T = 10$, which corresponds to 100 iterations. We emphasize that the results here are not sensitive to the choice of α, provided that it is taken to be sufficiently small. The range of suitable length-steps will, in general, depend on the strength of the nonlinearity of h and the dimension of the problem. For the choice of T, stopping criteria for convergence could be added to the algorithms. Here for simplicity we set T manually and let all algorithms run for a sufficient amount of time.

We plot the level curves of the posterior density of $u = (u_1, u_2)^T$ in Figure 5.1. The forward model is approximately linear, as can be seen from the contour plots in Figure 5.1 or directly from Equation (5.5.2). Hence it can be used to validate the claims we made in previous sections.

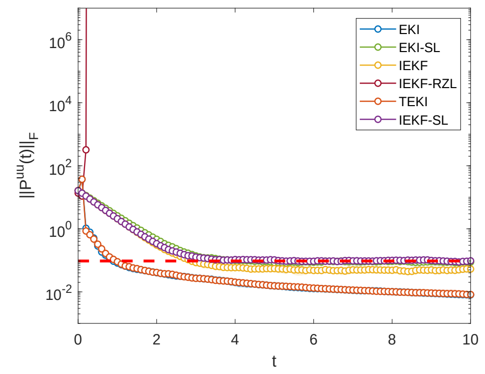


Figure 5.2: Evolution of the Frobenius norm of the ensemble covariance $P^{uu}(t)$. For reference, we also plot the Frobenius norm of the true posterior covariance (red dashed line). The norm of IEKF-RZL blows up after a few iterations. The norms of the EKI and TEKI are almost identical and monotonically decreasing. The norms of the new variants EKI-SL and IEKF-SL are similar and stabilize after around 40 iterations. The norm of IEKF lies between those of the old and new variants.

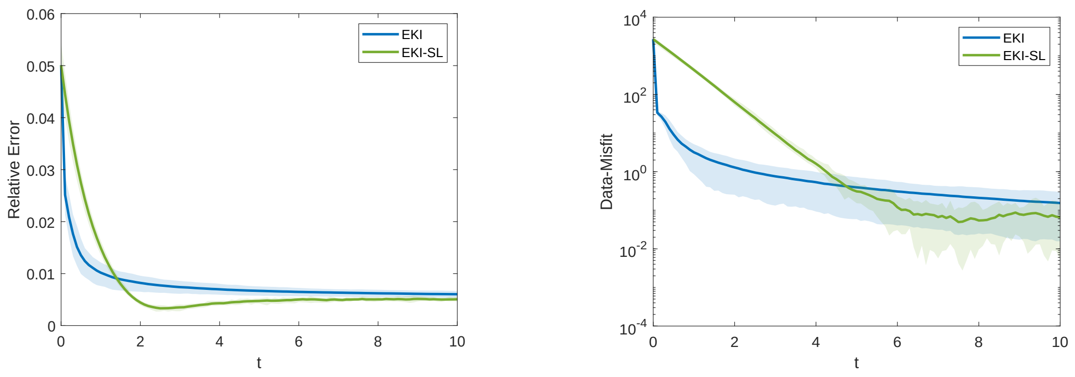


Figure 5.3: EKI & EKI-SL: Relative errors and data misfit w.r.t time t.

Figure 5.2 compares the time evolution of the empirical covariance of the different methods. The IEKF-RZL algorithm consistently blows up in the first few iterations due to the small noise which, as discussed in Section 5.3.1, is an important drawback. Due to the poor performance of IEKF-RZL in small noise regimes, we will not include this method in subsequent comparisons. The EKI and TEKI plots show the 'ensemble collapse' phenomenon

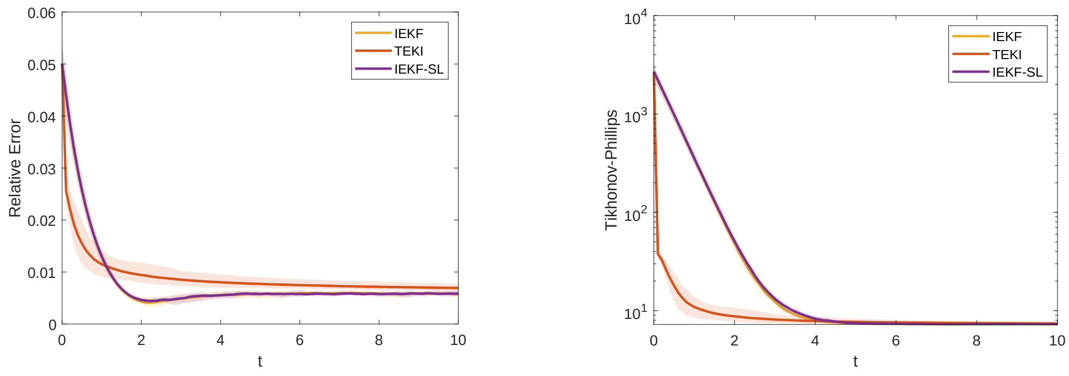


Figure 5.4: TEKI, IEKF & IEKF-SL: Relative errors and Tikhonov-Phillips objective w.r.t time t.

discussed in Sections 5.3.2 and 5.3.3. Note that the size of the empirical covariances of EKI and TEKI decreases monotonically, and by the 100-th iteration their size is one order of magnitude smaller than that of the new variants IEKF-SL and EKI-SL. The ensemble collapse of EKI and TEKI can also be seen in Figure 5.1, where we plot all the ensemble members after 100 iterations. In contrast, Figure 5.2 shows that the size of the empirical covariances of the new variants IEKF-SL and EKI-SL stabilizes after around 40 iterations, and Figure 5.1 suggests that the spread of the ensemble matches that of the posterior. These results are in agreement with the theoretical results derived in Sections 5.4.2 and 5.4.1 in a linear setting. Although we have not discussed the IEKF method when α is small, its ensemble covariance does not collapse in the linear setting, as can be seen in Figure 5.2.

In Figures 5.3 and 5.4 we show the performance of the different methods along the full iteration sequence. Here and in subsequent numerical examples we use two performance assessments. First, the relative error, defined as $|m(t) - u^\dagger|/|u^\dagger|$ where $m(t)$ is the ensemble's empirical mean, which evaluates how well the ensemble mean approximates the truth. Second, we assess how each ensemble method performs in terms of its own optimization objective. Precisely, we report the data-misfit objective $\mathsf{J}_{\mathrm{DM}}\big(m(t)\big)$ for EKI and EKI-SL, and the Tikhonov-Phillips objective $\mathsf{J}_{\mathrm{TP}}\big(m(t)\big)$ for IEKF, TEKI and IEKF-SL. We run 10 trials for each algorithm, using different ensemble initializations (drawn from the same prior), and generate the error bars accordingly. Since this is a simple toy problem, all methods perform well. However, we note that the first iterations of EKI and TEKI reduce the objective faster than other algorithms.

5.5.2 High-dimensional linear inverse problem

In this subsection we consider a linear Bayesian inverse problem from [Iglesias et al., 2013]. This example illustrates the use of iterative ensemble Kalman methods in settings where both the size of the ensemble and the dimension of the data are significantly smaller than the dimension of the unknown parameter.

Problem setup

Consider the one dimensional elliptic equation

$$\begin{aligned} -\frac{\mathrm{d}^2 p}{\mathrm{d}x^2} + p = u, \quad x \in (0,\pi), \\ p(0) = p(\pi) = 0. \end{aligned} \tag{5.5.4}$$

We seek to recover u from noisy observation of p at $k = 2^4 - 1$ equispaced points $x_j = \frac{j}{2^4}\pi$. We assume that the data is generated from the model

$$y_j = p(x_j) + \eta_j, \quad j = 1, \ldots, k, \tag{5.5.5}$$

where $\eta_j \sim \mathcal{N}(0, \gamma^2)$ are independent. By defining $A = -\frac{\mathrm{d}^2}{\mathrm{d}x^2} + id$ and letting $\mathcal{O}$ be the observation operator defined by $\big(\mathcal{O}(p)\big)_j = p(x_j)$, we can rewrite (5.5.5) as

$$y = h(u) + \eta, \quad \eta \sim \mathcal{N}(0, \gamma^2 I_k),$$

where $h = \mathcal{O} \circ A^{-1}$. The forward problem (5.5.4) is solved on a uniform mesh with mesh-width $w = 2^{-8}$ by a finite element method with continuous, piecewise linear basis functions. We assume that the unknown parameter u has a Gaussian prior distribution, $u \sim \mathcal{N}(0, C_0)$ with covariance operator $C_0 = 10(A - id)^{-1}$ with homogeneous Dirichlet boundary conditions. This prior can be interpreted as the law of a Brownian bridge between 0 and π. For computational purposes we view u as a random vector in $\mathbb{R}^{2^8}$ and the linear map $h(u)$ is represented by a matrix $H \in \mathbb{R}^{(2^4-1)\times 2^8}$. The true parameter $u^\dagger$ is sampled from this prior (cf. Figure 5.5), and is used to generate synthetic observation data $y = Hu^\dagger + \eta$ with noise level $\gamma = 0.01$.

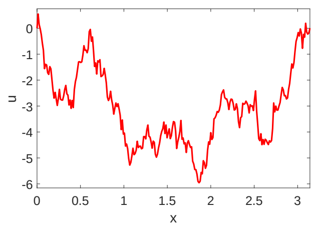
Implementation details and numerical results

We set the ensemble size to be $N = 50$. The initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is drawn independently from the prior. The length-step α is fixed to be 0.05 for all methods. We run each algorithm up to time $T = 30$, which corresponds to 600 iterations.

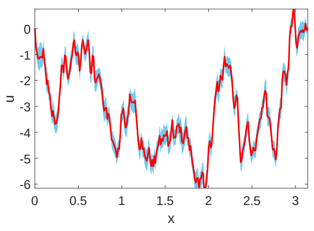
We note that the linear inverse problem considered here has input dimension $2^8 = 256$, which is much larger than the ensemble size N. Combining the results from Figure 5.5, 5.6 and 5.7, EKI and EKI-SL clearly overfit the data. In contrast, TEKI over-regularizes the data, and we easily notice the ensemble collapse. The IEKF and IEKF-SL lie in between, and approximate the truth slightly better.

5.5.3 Lorenz-96 model

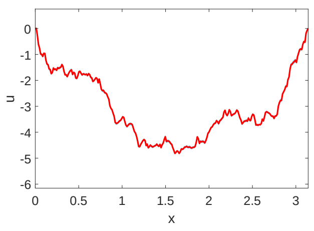
In this subsection we investigate the use of iterative ensemble Kalman methods to recover the initial condition of the Lorenz-96 system from partial and noisy observation of the solution at two positive times. The experimental framework is taken from [Chada and Tong, 2022] and is illustrative of the use of iterative ensemble Kalman methods in numerical weather forecasting.



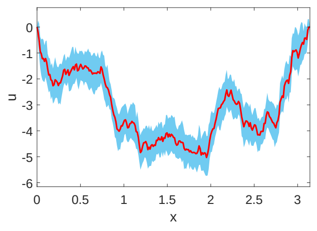
(a) Truth $u^\dagger$.



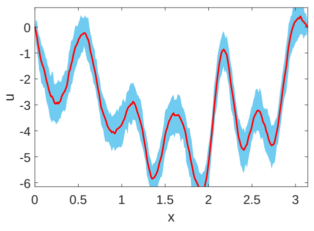
(b) EKI.



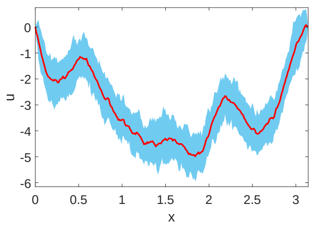
(c) TEKI.



(d) IEKF.

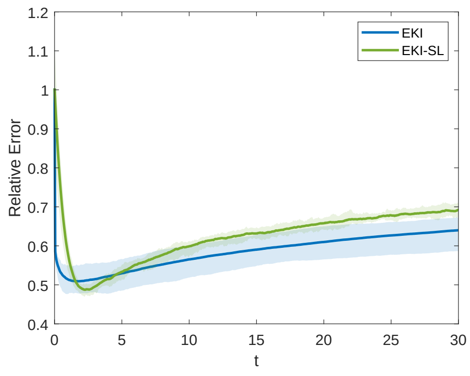


(e) EKI-SL.



(f) IEKF-SL.

Figure 5.5: Ensemble mean (red) at the final iteration, with 10, 90-quantiles (blue).



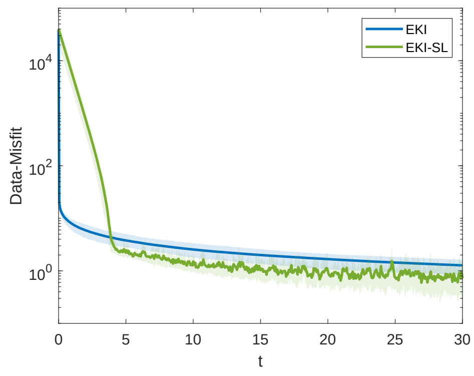


Figure 5.6: EKI & EKI-SL: Relative errors and data misfit w.r.t time t.

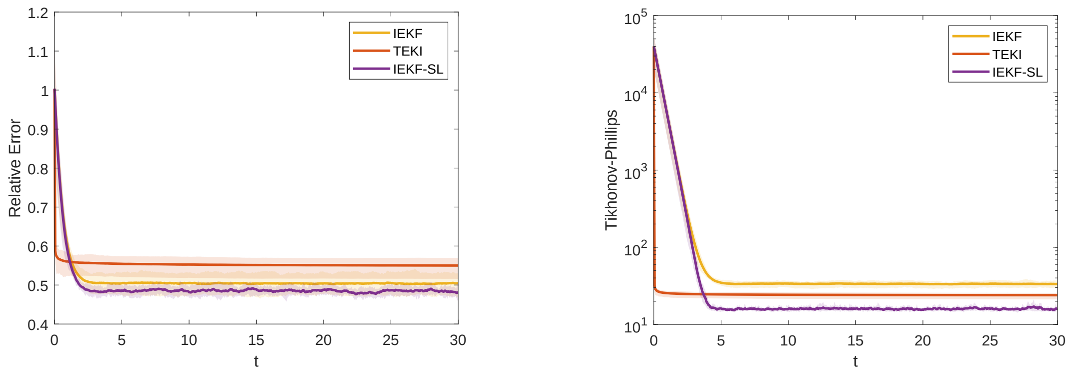


Figure 5.7: IEKF, TEKI & IEKF-SL: Relative errors and Tikhonov-Phillips objective w.r.t time t.

Problem setup

Consider the dynamical system

$$\begin{gathered}\frac{dz_l}{dt} = z_{l-1}(z_{l+1} - z_{l-2}) - z_l + F, \quad l = 1, \dots, d, \\ z_0 = z_d, \quad z_{d+1} = z_1, \quad z_{-1} = z_{d-1}.\end{gathered} \tag{5.5.6}$$

Here z_l denotes the l^{th} coordinate of the current state of the system and F is a constant forcing with default value of $F = 8$. The dimension d is often chosen as 40. We want to recover the initial condition

$$u := (z_1(0), \dots, z_d(0))^T$$

of (5.5.6) from noisy partial measurements at discrete times $\{s_i\}_{i=1}^{I}$:

$$y_{i,j} = u_{l_j}(s_i) + \eta_{i,j}, \tag{5.5.7}$$

where $\{l_j\}_{j=1}^{J} \subset \{1, 2, \dots, d\}$ is the subset of observed coordinates, and $\eta_{i,j} \sim \mathcal{N}(0, \gamma^2)$ are assumed to be independent. In our numerical experiments we set $I = 2$, $J = 20$, $\{s_1, s_2\} = \{0.3, 0.6\}$, $\{l_j\}_{j=1}^{20} = \{1, 3, 5, \dots, 39\}$. We set the prior on u to be a Gaussian $\mathcal{N}(0, 2I_d)$. The true parameter $u^\dagger \in \mathbb{R}^{40}$ is shown in Figure 5.8, and is used to generate the observation data $\{y_{i,j}\}$ according to (5.5.7) with noise level $\gamma = 0.01$.

Implementation details and numerical results

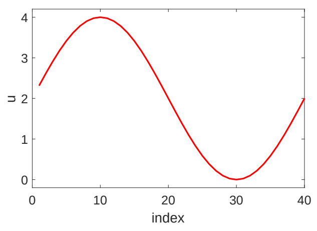
We set the ensemble size to be $N = 50$. The initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is drawn independently from the prior. The length-step α is fixed to be 0.05 for all methods. We run each algorithm up to time $T = 30$, which corresponds to 600 iterations.

This is a moderately high dimensional nonlinear problem, where the forward model involves a black-box solver of differential equations. Figure 5.8 clearly indicates an ensemble collapse for the EKI and TEKI methods. Although they are still capable of finding a descending direction of the loss direction (see Figures 5.9 and 5.10), iterates get stuck in a local minimum. In contrast, we can see the advantage of IEKF, EKI-SL and TEKI-SL in this setting: intuitively, a broader spread of the ensemble allows these methods to 'explore' different regions in the input space, and thereby to find a solution with potentially lower loss.

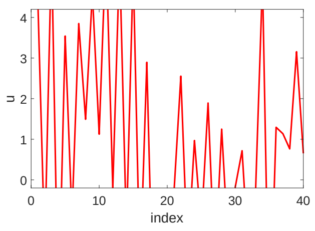
We notice that in practice 'covariance inflation' is often applied in EKI and TEKI algorithms, by manually injecting small random noise after each ensemble update. In general, this technique will 'force' a non-zero empirical covariance, prevent ensemble collapse and boost the performence of EKI and TEKI. However, the amount of noise injected is an additional hyperparameter that should be chosen manually, which should depend on the input dimension, observation noise, etc. Here we give a fair comparison of the different methods introduced, under the same time-stepping setting, with as few hyperparameters as possible.

5.5.4 High-dimensional nonlinear regression

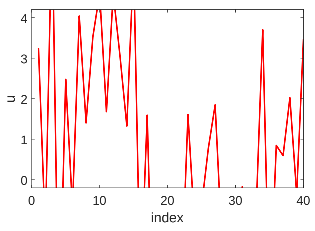
In this subsection we consider a nonlinear regression problem with a highly oscillatory forward map introduced in [Haber et al., 2018], where the authors investigate the use of iterative



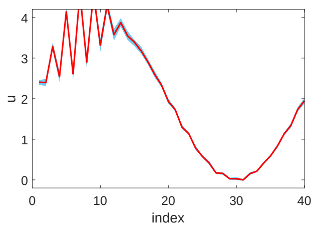
(a) Truth $u^\dagger$.



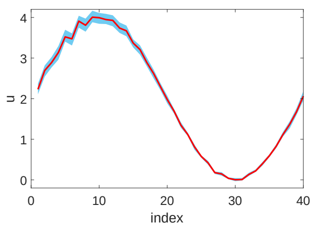
(b) EKI.



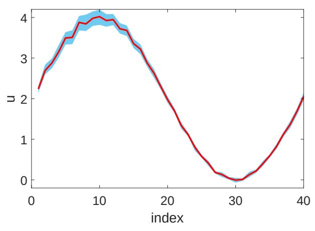
(c) TEKI.



(d) IEKF.



(e) EKI-SL.



(f) IEKF-SL.

Figure 5.8: Ensemble mean (red) at the final iteration, with 10, 90-quantiles (blue).

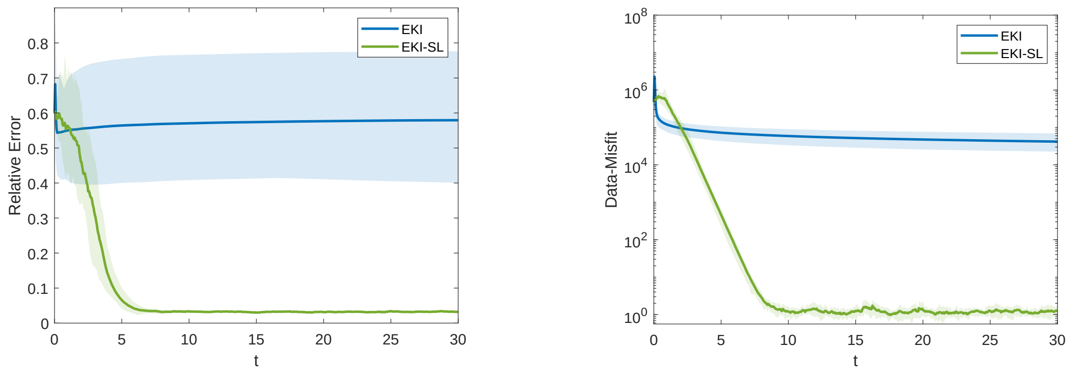


Figure 5.9: EKI & EKI-SL: Relative errors and data misfit w.r.t time t.

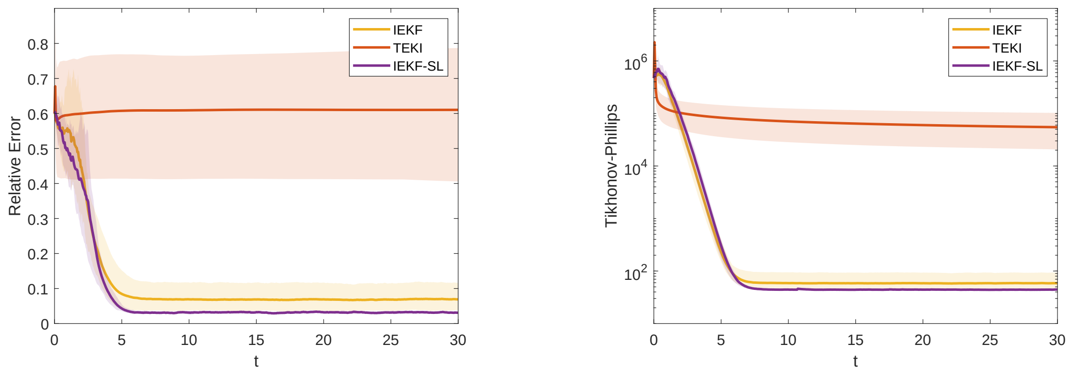


Figure 5.10: IEKF, TEKI & IEKF-SL: Relative errors and Tikhonov-Phillips objective w.r.t time t.

ensemble Kalman methods to train neural networks without back propagation.

Problem setup

We consider a nonlinear regression problem

$$y = h(u) + \eta, \quad \eta \sim \mathcal{N}(0, \gamma^2 I_k),$$

where $u \in \mathbb{R}^d, y \in \mathbb{R}^k$, and h is defined by

$$h(u) := Au + \sin(cBu), \tag{5.5.8}$$

where $A, B \in \mathbb{R}^{k\times d}$ are random matrices with independent $\mathcal{N}(0,1)$ entries. We set $d = 200$, $k = 150$ and $c = 20$. We want to recover u from y. We assume that the unknown u has a Gaussian prior $u \sim \mathcal{N}(0, 4I_d)$. The true parameter $u^\dagger$ is set to be $2 \cdot \mathbf{1}$, where $\mathbf{1}$ is the all-one vector. Observation data is generated as $y = h(u^\dagger) + \eta$.

By definition, h is highly oscillatory, and we may expect that the loss function, either Tikhonov-Phillips or data misfit objective, will have many local minima. Figure 5.11 visualizes the Tihonov-Phillips objective function $\mathsf{J}_{\text{TP}}(u)$ with respect to two randomly choosen coordinates while other coordinates are fixed to value of 2. The data misfit objective exhibits a similar behavior.

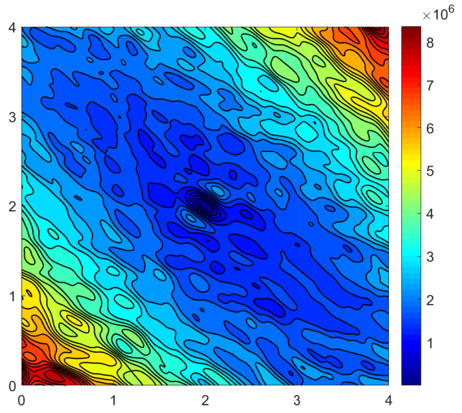
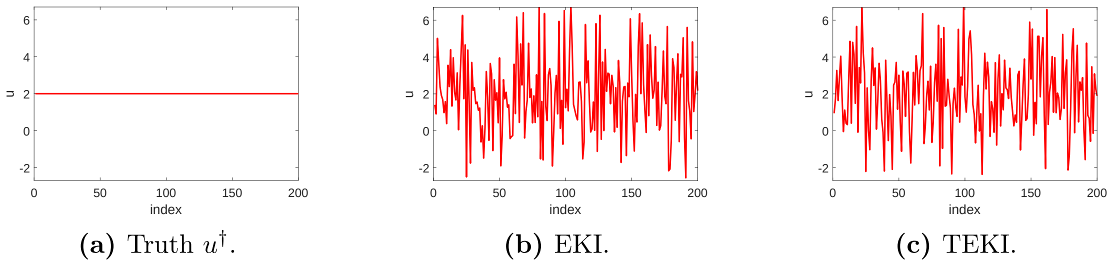
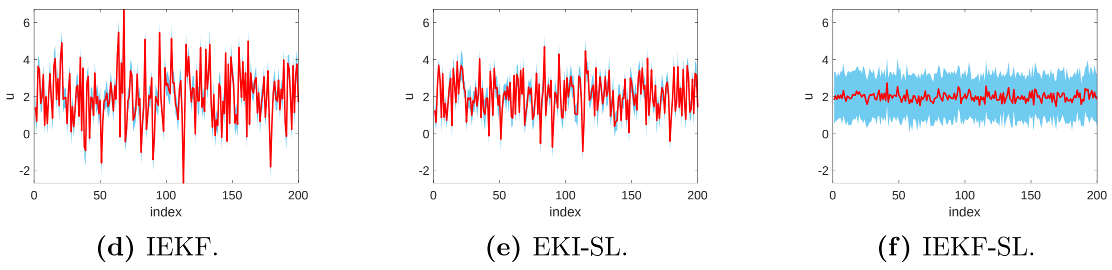


Figure 5.11: Tikhonov-Phillips objective function with respect to two randomly chosen coordinates.

Implementation details and numerical results

We set the ensemble size to be $N = 200$. The initial ensemble $\{u_0^{(n)}\}_{n=1}^N$ is drawn independently from the prior. The length-step α is fixed to be 0.05 for all methods. We set $\gamma = 0.01$ for the observation noise. We run each algorithm up to time $T = 30$, which corresponds to 600 iterations.

We notice that this is a difficult problem, due to its high dimensionality and nonlinearity. All methods except for IEKF-SL are not capable of reconstructing the truth. In particular, from Figures 5.12, 5.13 and 5.14, both EKI and TEKI do a poor job with relative error larger than 1, while IEKF and EKI-SL have slightly better performance. It is worth noticing from Figure 5.14 that IEKF-SL has a larger Tikhonov-Phillips objective function, with a much lower relative error. This may suggest that other types of regularization objectives can be used, other that the Tikhonov-Phillips objective, to solve this problem.

(a) Truth $u^\dagger$. (b) EKI. (c) TEKI. (d) IEKF. (e) EKI-SL. (f) IEKF-SL.

Figure 5.12: Ensemble mean (red) at the final iteration, with 10, 90-quantiles (blue).

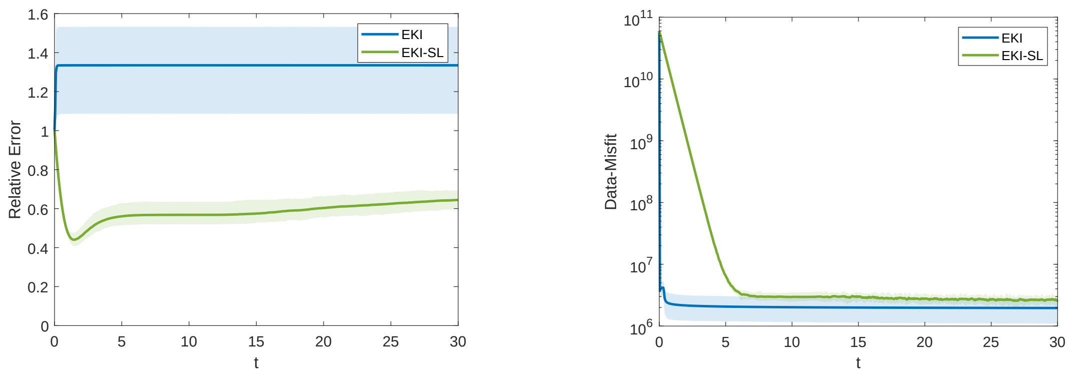


Figure 5.13: EKI & EKI-SL: Relative errors and data misfit w.r.t time t.

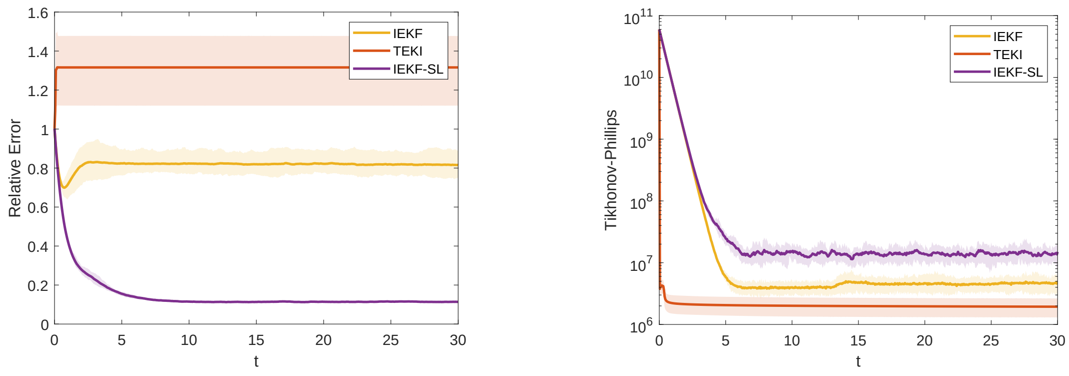


Figure 5.14: IEKF, TEKI & IEKF-SL: Relative errors and Tikhonov-Phillips objective w.r.t time t.

5.6 Conclusions and open directions

In this paper we have provided a unified perspective of iterative ensemble Kalman methods and introduced some new variants. These new variants include a statistical linearization of both EKI and the IEnKF. Our numerical experiments suggest that the IEnKF-SL does not suffer from the overfitting of data. Furthermore and more interestingly, that for high-dimensional nonlinear problems, the IEnKF-SL may outperform other known methodologies. This is a promising result which has potential for other highly nonlinear inverse problems. However, for linear inverse problems, all variants discussed, new and known, perform well. As stated the advantage of such new variants is that no parameter tuning is required. We hope that our work will stimulate further research in this active area, and we conclude with a list of some open directions.

- Continuum limits have been formally derived in our work. The rigorous derivation and analysis of SDE continuum limits, possibly in nonlinear settings, deserves further research.
- We have advocated the analysis of continuum limits for the understanding and design of iterative ensemble Kalman methods, but it would also be desirable to develop a framework for the analysis of discrete, implementable algorithms, and to further un-

derstand the potential benefits of various discretizations of a given continuum SDE system.

- From a theoretical viewpoint, it would be desirable to further analyze the convergence and stability of iterative ensemble methods with small or moderate ensemble size. While mean-field limits can be revealing, in practice the ensemble size is often not sufficiently large to justify the mean-field assumption. It would also be important to further analyze these questions in mildly nonlinear settings.

- An important methodological question, still largely unresolved, is the development of adaptive and easily implementable line search schemes and stopping criteria for ensemble-based optimization schemes. An important work in this direction is [Iglesias and Yang, 2021].

- Another avenue for future methodological research is the development of iterative ensemble Kalman methods that are sparse-promoting, considering alternative regularizations beyond the least-squares objectives discussed in our paper [Kovachki and Stuart, 2019; Lee, 2021; Schneider et al., 2022].

- Ensemble methods are cheap in comparison to derivative-based optimization methods and Markov chain Monte Carlo sampling algorithms. It is thus natural to use ensemble methods to build ensemble preconditioners or surrogate models to be used within more expensive but accurate computational approaches.

- Finally, a broad area for further work is the application of iterative ensemble Kalman filters to new problems in science and engineering.

REFERENCES

S. I. Aanonsen, G. Nœvdal, D. S. Oliver, A. C. Reynolds, and B. Vallès. The ensemble Kalman filter in reservoir engineering—a review. *Spe Journal*, 14(03):393–412, 2009.

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. M. Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32(3):405–431, 2017.

D. J. Albers, P.-A. Blancquart, M. E. Levine, E. E. Seylabi, and A. M. Stuart. Ensemble Kalman methods with constraints. *Inverse Problems*, 2019.

B. D. Anderson and J. B. Moore. *Optimal filtering*. Courier Corporation, 2012.

J. L. Anderson. An ensemble adjustment Kalman filter for data assimilation. *Monthly Weather Review*, 129(12):2884–2903, 2001.

J. L. Anderson and S. L. Anderson. A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly Weather Review*, 127 (12):2741–2758, 1999.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari. Learning dynamical systems from partial observations. *arXiv preprint arXiv:1902.11136*, 2019.

R. Becker and B. Vexler. Mesh refinement and numerical sensitivity analysis for parameter calibration of partial differential equations. *Journal of Computational Physics*, 206(1): 95–110, 2005.

B. M. Bell. The iterated Kalman smoother as a Gauss–Newton method. *SIAM Journal on Optimization*, 4(3):626–636, 1994.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

T. Bengtsson, P. Bickel, B. Li, et al. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. *Probability and statistics: Essays in honor of David A. Freedman*, 2:316–334, 2008.

P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

A. Beskos, G. O. Roberts, A. M. Stuart, and J. Voss. MCMC methods for diffusion bridges. *Stochastics and Dynamics*, 8(03):319–350, 2008.

D. Bigoni, O. Zahm, A. Spantini, and Y. M. Marzouk. Greedy inference with layers of lazy maps. *arXiv preprint arXiv:1906.00031*, 2019.

D. Bigoni, Y. Chen, N. G. Trillos, Y. Marzouk, and D. Sanz-Alonso. Data-driven forward discretizations for Bayesian inversion. *Inverse Problems*, 36(10):105008, 2020.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

C. Blömker, D.and Schillings, P. Wacker, and S. Weissmann. Well posedness and convergence analysis of the ensemble Kalman inversion. *Inverse Problems*, 2019.

P. J. Blonigan and Q. Wang. Least squares shadowing sensitivity analysis of a modified Kuramoto–Sivashinsky equation. *Chaos, Solitons & Fractals*, 64:16–25, 2014. Nonequilibrium Statistical Mechanics: Fluctuations and Response.

D. Bloömker, C. Schillings, and P. Wacker. A strongly convergent numerical scheme from ensemble Kalman inversion. *SIAM Journal on Numerical Analysis*, 56(4):2537–2562, 2018.

M. Bocquet, C. A. Pires, and L. Wu. Beyond Gaussian statistical modeling in geophysical data assimilation. *Monthly Weather Review*, 138(8):2997–3023, 2010.

M. Bocquet, J. Brajard, A. Carrassi, and L. Bertino. Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models. *Nonlinear Processes in Geophysics*, 26(3):143–162, 2019.

M. Bocquet, J. Brajard, A. Carrassi, and L. Bertino. Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization. *arXiv preprint arXiv:2001.06270*, 2(1):55–80, 2020.

L. Borcea, V. Druskin, and L. Knizhnerman. On the continuum limit of a discrete inverse spectral problem on optimal finite difference grids. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 58(9): 1231–1279, 2005.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

J. Brajard, A. Carrassi, M. Bocquet, and L. Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model. *Journal of Computational Science*, 44:101171, 2020.

S. L. Brunton and J. N. Kutz. *Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* Cambridge University Press, 2019.

S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

J. M. Burgers. A mathematical model illustrating the theory of turbulence. *Advances in applied mechanics*, 1:171–199, 1948.

A. Carrassi, M. Ghil, A. Trevisan, and F. Uboldi. Data assimilation as a nonlinear dynamical systems problem: Stability and convergence of the prediction-assimilation system. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 18(2):023112, 2008.

A. Carrassi, M. Bocquet, A. Hannart, and M. Ghil. Estimating model evidence using data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 143(703):866–880, 2017.

N. K. Chada. Analysis of hierarchical ensemble Kalman inversion. *arXiv preprint arXiv:1801.00847*, 2018.

N. K. Chada and X. Tong. Convergence acceleration of ensemble Kalman inversion in nonlinear settings. *Mathematics of Computation*, 91(335):1247–1280, 2022.

N. K. Chada, M. Iglesias, L. Roininen, and A. M. Stuart. Parameterizations for ensemble Kalman inversion. *Inverse Problems*, 34(2018), 2017.

N. K. Chada, C. Schillings, and S. Weissmann. On the incorporation of box-constraints for ensemble Kalman inversion. *arXiv preprint arXiv:1908.00696*, 2019.

N. K. Chada, A. M. Stuart, and X. T. Tong. Tikhonov regularization within ensemble Kalman inversion. *SIAM Journal on Numerical Analysis*, 58(2):1263–1294, 2020.

N. K. Chada, Y. Chen, and D. Sanz-Alonso. Iterative ensemble Kalman methods: A unified perspective with some new variants. *Foundations of Data Science*, 3(3):331–369, 2021.

K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

N. Chen and A. J. Majda. Conditional Gaussian systems for multiscale nonlinear stochastic systems: prediction, state estimation and uncertainty quantification. *Entropy*, 20(7), 2018.

N. Chen and D. Qi. A physics-informed data-driven algorithm for ensemble forecast of complex turbulent systems. *arXiv preprint arXiv:2204.08547*, 2022.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31, pages 6571–6583, 2018.

Y. Chen and D. S. Oliver. Ensemble randomized maximum likelihood method as an iterative ensemble smoother. *Mathematical Geosciences*, 44:1–26, 2012.

Y. Chen, D. Sanz-Alonso, and R. Willett. Autodifferentiable Ensemble Kalman Filters. *SIAM Journal on Mathematics of Data Science*, 4(2):801–833, 2022.

Y. Chen, D. Sanz-Alonso, and R. Willett. Reduced-Order Autodifferentiable Ensemble Kalman Filters. *arXiv preprint arXiv:2301.11961*, 2023.

J. A. Christen and C. Fox. Markov chain Monte Carlo using an approximation. *Journal of Computational and Graphical statistics*, 14(4):795–810, 2005.

J. Cockayne, C. J. Oates, T. J. Sullivan, and M. Girolami. Bayesian probabilistic numerical methods. *SIAM Review*, 61(4):756–789, 2019.

T. J. Cocucci, M. Pulido, M. Lucini, and P. Tandeo. Model error covariance estimation in particle and ensemble Kalman filters using an online expectation–maximization algorithm. *Quarterly Journal of the Royal Meteorological Society*, 147(734):526–543, 2021.

A. Corenflos, J. Thornton, A. Doucet, and G. Deligiannidis. Differentiable Particle Filtering via Entropy-Regularized Optimal Transport. *arXiv preprint arXiv:2102.07850*, pages 2100–2111, 2021.

S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC methods for functions: modifying old algorithms to make them faster. *Statistical Science*, 28(3):424–446, 2013.

D. Crisan, P. Del Moral, A. Jasra, and H. Ruzayqat. Log-normalization constant estimation using the ensemble Kalman-Bucy filter with application to high-dimensional models. *arXiv preprint arXiv:2101.11460*, 2021.

T. Cui, Y. M. Marzouk, and K. E. Willcox. Data-driven model reduction for the Bayesian solution of inverse problems. *International Journal for Numerical Methods in Engineering*, 102(5):966–990, 2015.

T. Cui, K. J. H. Law, and Y. M. Marzouk. Dimension-independent likelihood-informed MCMC. *Journal of Computational Physics*, 304:109–137, 2016.

E. De Brouwer, J. Simm, A. Arany, and Y. Moreau. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. *arXiv preprint arXiv:1905.12374*, 32, 2019.

P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications.* Series: Probability and Applications, Springer-Verlag, New York, 2004.

P. Del Moral, J. Tugaut, et al. On the stability and the uniform propagation of chaos properties of ensemble Kalman–Bucy filters. *The Annals of Applied Probability*, 28(2): 790–850, 2018.

C. Dellacherie and P.-A. Meyer. *Probabilities and Potentials: Potential theory for Discrete and Continuous Semigroups*. Elsevier, 2011.

T. DelSole and X. Yang. State and parameter estimation in stochastic dynamical models. *Physica D: Nonlinear Phenomena*, 239(18):1781–1788, 2010.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

J. E. Dennis Jr and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

L. Dieci and E. S. Van Vleck. Lyapunov and Sacker–Sell spectral intervals. *Journal of Dynamics and Differential Equations*, 19(2):265–293, 2007.

L. Dieci and E. S. Van Vleck. Lyapunov exponents: Computation. *Encyclopedia of Applied and Computational Mathematics*, pages 834–838, 2015.

Z. Ding and Q. Li. Ensemble Kalman sampler: Mean-field limit and convergence analysis. *SIAM Journal on Mathematical Analysis*, 53(2):1546–1578, 2021.

A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

D. Dreano, P. Tandeo, M. Pulido, B. Ait-El-Fquih, T. Chonavel, and I. Hoteit. Estimating model-error covariances in nonlinear state-space models using Kalman smoothing and the expectation–maximization algorithm. *Quarterly Journal of the Royal Meteorological Society*, 143(705):1877–1885, 2017.

C. Drovandi, R. G. Everitt, A. Golightly, D. Prangle, et al. Ensemble MCMC: accelerating pseudo-marginal MCMC for state space models using the ensemble Kalman filter. *Bayesian Analysis*, 17(1):223–260, 2021.

Q. Du and M. Gunzburger. Grid generation and optimization based on centroidal Voronoi tessellations. *Applied mathematics and computation*, 133(2-3):591–607, 2002.

Y. Efendiev, T. Hou, and W. Luo. Preconditioning Markov chain Monte Carlo simulations using coarse-scale models. *SIAM Journal on Scientific Computing*, 28(2):776–803, 2006.

A. A. Emerick and A. C. Reynolds. Ensemble smoother with multiple data assimilation. *Computers & Geosciences*, 55:3–15, 2013.

O. G. Ernst, B. Sprungk, and H.-J. Starkloff. Analysis of the ensemble and polynomial chaos Kalman filters in Bayesian inverse problems. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):823–851, 2015.

G. Evans and P. V. Leeuwen. Assimilation of Geosat altimeter data for the Agulhas current using the ensemble Kalman filter with a quasigeostrophic model. *Monthly Weather Review*, 124(1):85–96, 1996.

G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

G. Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

G. Evensen, F. C. Vossepoel, and P. J. van Leeuwen. *EnKF with the Lorenz Equations*, pages 151–156. Springer International Publishing, 2022.

A. Farchi, P. Laloyaux, M. Bonavita, and M. Bocquet. Using machine learning to correct model error in data assimilation and forecast applications. *Quarterly Journal of the Royal Meteorological Society*, 147(739):3067–3084, 2021.

M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *arXiv preprint arXiv:1710.05741*, 2017.

M. Frangos, Y. Marzouk, K. Willcox, and B. van Bloemen Waanders. Surrogate and reduced-order modeling: A comparison of approaches for large-scale statistical inverse problems. *Large-Scale Inverse Problems and Quantification of Uncertainty*, pages 123–149, 2010.

R. Furrer and T. Bengtsson. Estimation of high-dimensional prior and posterior covariance matrices in Kalman filter variants. *Journal of Multivariate Analysis*, 98(2):227–255, 2007.

A. Garbuno-Inigo, F. Hoffmann, W. Li, and A. M. Stuart. Interacting Langevin diffusions: Gradient structure and ensemble Kalman sampler. *SIAM Journal on Applied Dynamical Systems*, 19(1):412–441, 2020.

N. García Trillos and D. Sanz-Alonso. The Bayesian formulation and well-posedness of fractional elliptic inverse problems. *Inverse Problems*, 33(6):065006, 2017.

N. Garcia Trillos and D. Sanz-Alonso. Continuum limits of posteriors in graph Bayesian inverse problems. *SIAM Journal on Mathematical Analysis*, 50(4):4020–4040, 2018.

N. Garcia Trillos, Z. Kaplan, T. Samakhoana, and D. Sanz-Alonso. On the consistency of graph-based Bayesian learning and the scalability of sampling algorithms. *arXiv preprint arXiv:1710.07702*, 21(28):1–47, 2017.

G. Gaspari and S. E. Cohn. Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125(554):723–757, 1999.

O. A. Ghattas and D. Sanz-Alonso. Non-Asymptotic Analysis of Ensemble Kalman Updates: Effective Dimension and Localization. *arXiv preprint arXiv:2208.03246*, 2022.

M. B. Giles. Collected matrix derivative results for forward and reverse mode algorithmic differentiation. In *Advances in Automatic Differentiation*, pages 35–44. Springer, 2008a.

M. B. Giles. Multilevel Monte Carlo path simulation. *Operations Research*, 56(3):607–617, 2008b.

F. J. Gonzalez and M. Balajewicz. Deep Convolutional Recurrent Autoencoders for Learning Low-dimensional Feature Dynamics of Fluid Systems. *arXiv preprint arXiv:1808.01346*, 2018.

N. J. Gordon, D. J. Salmond, and A. F. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE proceedings F (radar and signal processing)*, 140(2):107–113, 1993.

G. A. Gottwald and S. Reich. Supervised learning from noisy observations: Combining machine-learning techniques with data assimilation. *Physica D: Nonlinear Phenomena*, 423:132911, 2021.

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

P. J. Green and A. Mira. Delayed rejection in reversible jump Metropolis–Hastings. *Biometrika*, 88(4):1035–1053, 2001.

I. Grooms. A note on the formulation of the Ensemble Adjustment Kalman Filter. *arXiv preprint arXiv:2006.02941*, 2020.

Y. Gu and D. S. Oliver. An iterative ensemble Kalman filter for multiphase fluid flow data assimilation. *Spe Journal*, 12(04):438–446, 2007.

J. Guibas, M. Mardani, Z. Li, A. Tao, A. Anandkumar, and B. Catanzaro. Efficient token mixing for transformers via adaptive Fourier neural operators. In *International Conference on Learning Representations*, 2021.

P. A. Guth, C. Schillings, and S. Weissmann. Ensemble Kalman filter for neural network-based one-shot inversion. *Optimization and Control for Partial Differential Equations: Uncertainty Quantification, Open and Closed-Loop Control, and Shape Optimization*, 29: 393, 2022.

E. Haber, F. Lucka, and L. Ruthotto. Never look back-A modified EnKF method and its application to the training of neural networks without back propagation. *arXiv preprint arXiv:1805.08034*, 2018.

M. Hairer, A. M. Stuart, and J. Voss. Signal processing problems on function space: Bayesian formulation, stochastic PDEs and effective MCMC methods. *The Oxford Handbook of Nonlinear Filtering*, pages 833–873, 2011.

T. M. Hamill, J. S. Whitaker, and C. Snyder. Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. *Monthly Weather Review*, 129 (11):2776–2790, 2001.

M. Hanke. A regularizing Levenberg-Marquardt scheme, with applications to inverse groundwater filtration problems. *Inverse problems*, 13(1):79, 1997.

A. Hannart, A. Carrassi, M. Bocquet, M. Ghil, P. Naveau, M. Pulido, J. Ruiz, and P. Tandeo. DADA: Data assimilation for the detection and attribution of weather and climate-related events. *Climatic Change*, 136(2):155–174, 2016.

J. Harlim, D. Sanz-Alonso, and R. Yang. Kernel methods for Bayesian elliptic inverse problems on manifolds. *SIAM/ASA Journal on Uncertainty Quantification*, 8(4):1414–1445, 2020.

J. Harlim, S. W. Jiang, S. Liang, and H. Yang. Machine learning for prediction with missing dynamics. *Journal of Computational Physics*, 428:109922, 2021.

Y. He, S.-H. Kang, W. Liao, H. Liu, and Y. Liu. Robust identification of differential equations by numerical techniques from a single set of noisy observation. *SIAM Journal on Scientific Computing*, 44(3):A1145–A1175, 2022a.

Y. He, N. Suh, X. Huo, S. H. Kang, and Y. Mei. Asymptotic theory of regularized PDE identification from a single noisy trajectory. *SIAM/ASA Journal on Uncertainty Quantification*, 10(3):1012–1036, 2022b.

Y. He, H. Zhao, and Y. Zhong. How much can one learn a partial differential equation from its solution? *arXiv preprint arXiv:2204.04602*, 2022c.

M. Herty and G. Visconti. Kinetic methods for inverse problems. *arXiv preprint arXiv:1811.09387*, 2018.

P. L. Houtekamer and H. L. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 126(3):796–811, 1998.

P. L. Houtekamer and H. L. Mitchell. A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129(1):123–137, 2001.

P. L. Houtekamer and F. Zhang. Review of the ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 144(12):4489–4532, 2016.

D. Z. Huang, T. Schneider, and A. M. Stuart. Iterated Kalman methodology for inverse problems. *Journal of Computational Physics*, 463:111262, 2022.

M. Iglesias and Y. Yang. Adaptive regularisation for ensemble Kalman inversion. *Inverse Problems*, 37(2):025008, 2021.

M. A. Iglesias. A regularizing iterative ensemble Kalman method for PDE-constrained inverse problems. *Inverse Problems*, 32(2):025002, 2016.

M. A. Iglesias, K. J. Law, and A. M. Stuart. Ensemble Kalman methods for inverse problems. *Inverse Problems*, 29(4):045001, 2013.

T. Ishizone, T. Higuchi, and K. Nakamura. Ensemble Kalman variational objectives: nonlinear latent trajectory inference with a hybrid of variational inference and ensemble Kalman filter. *arXiv preprint arXiv:2010.08729*, 2020.

M. Jardak, I. Navon, and M. Zupanski. Comparison of sequential data assimilation methods for the Kuramoto–Sivashinsky equation. *International journal for numerical methods in fluids*, 62(4):374–402, 2010.

A. H. Jazwinski. *Stochastic processes and filtering theory.* Courier Corporation, 2007.

I. D. Jordan, P. A. Sokół, and I. M. Park. Gated Recurrent Units Viewed Through the Lens of Continuous Time Dynamical Systems. *Frontiers in Computational Neuroscience*, 15, 2021.

J. Kaipio and E. Somersalo. *Statistical and computational inverse problems*, volume 160. Springer Science & Business Media, 2006.

J. Kaipio and E. Somersalo. Statistical inverse problems: discretization, model reduction and inverse crimes. *Journal of computational and applied mathematics*, 198(2):493–504, 2007.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 03 1960.

M. Katzfuss, J. R. Stroud, and C. K. Wikle. Understanding the ensemble Kalman filter. *The American Statistician*, 70(4):350–357, 2016.

M. Katzfuss, J. R. Stroud, and C. K. Wikle. Ensemble Kalman methods for high-dimensional hierarchical dynamic space-time models. *Journal of the American Statistical Association*, 115(530):866–885, 2020.

D. Kelly and A. M. Stuart. Well-posedness and accuracy of the ensemble Kalman filter in discrete and continuous time. *Nonlinearity*, 27(10), 2014.

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926*, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations*, 2014.

A. Kloss, G. Martius, and J. Bohg. How to train your differentiable filter. *Autonomous Robots*, pages 1–18, 2021.

N. B. Kovachki and A. M. Stuart. Ensemble Kalman inversion: A derivative-free technique for machine learning tasks. *Inverse Problems*, 2019.

R. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Y. Kuramoto and T. Tsuzuki. Persistent propagation of concentration waves in dissipative media far from thermal equilibrium. *Progress of Theoretical Physics*, 55(2):356–369, 1976.

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

K. J. Law and A. M. Stuart. Evaluating data assimilation algorithms. *Monthly Weather Review*, 140(11):3757–3782, 2012.

K. J. Law, A. M. Stuart, and K. Zygalakis. Data Assimilation. *Cham, Switzerland: Springer*, 214, 2015.

K. J. Law, D. Sanz-Alonso, A. Shukla, and A. M. Stuart. Filter accuracy for the Lorenz 96 model: Fixed versus adaptive observation operators. *Physica D: Nonlinear Phenomena*, 325:1–13, 2016a.

K. J. Law, H. Tembine, and R. Tempone. Deterministic mean-field ensemble Kalman filtering. *SIAM Journal on Scientific Computing*, 38(3):A1251–A1279, 2016b.

W. G. Lawson and J. A. Hansen. Implications of stochastic and deterministic filters as ensemble-based data assimilation methods in varying regimes of error growth. *Monthly weather review*, 132(8):1966–1981, 2004.

T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. Auto-Encoding sequential Monte Carlo. In *International Conference on Learning Representations*, 2018.

F. Le Gland and L. Mevel. Recursive identification in hidden Markov models. In *Proceedings of the 36th Conference on Decision and Control, San Diego 1997*, volume 4, pages 3468–3473, 1997.

F. Le Gland, V. Monbet, and V.-D. Tran. *Large sample asymptotics for the ensemble Kalman filter*. PhD thesis, INRIA, 2009.

M. Lechner and R. Hasani. Learning Long-Term Dependencies in Irregularly-Sampled Time Series. *arXiv preprint arXiv:2006.04418*, 2020.

Y. Lee. l_p regularization for ensemble Kalman inversion. *SIAM Journal on Scientific Computing*, 43(5):A3417–A3437, 2021.

M. E. Levine and A. M. Stuart. A framework for machine learning of model error in dynamical systems. *arXiv preprint arXiv:2107.06658*, 2(07):283–344, 2021.

G. Li and A. C. Reynolds. An iterative ensemble Kalman filter for data assimilation. In *SPE annual technical conference and exhibition*. Society of Petroleum Engineers, 2007.

H. Li, V. V. Garg, and K. Willcox. Model adaptivity for goal-oriented inference using adjoints. *Computer Methods in Applied Mechanics and Engineering*, 331:1–22, 2018.

J. Li and Y. M. Marzouk. Adaptive construction of surrogates for the Bayesian solution of inverse problems. *SIAM Journal on Scientific Computing*, 36(3):A1163–A1186, 2014.

X. Li, T.-K. L. Wong, R. T. Chen, and D. Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 3870–3882. PMLR, 2020a.

Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020b.

C. Lieberman, K. Willcox, and O. Ghattas. Parameter and state model reduction for large-scale statistical inverse problems. *SIAM Journal on Scientific Computing*, 32(5):2523–2542, 2010.

Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

A. C. Lorenc. Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 112(474):1177–1194, 1986.

R. J. Lorentzen, K. K. Fjelde, J. Frøyen, A. C. Lage, G. Nævdal, and E. H. Vefring. Underbalanced drilling: Real time data interpretation and decision support. In *SPE/IADC drilling conference*. OnePetro, 2001.

E. N. Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on Predictability*, 1996.

S. Lu and S. V. Pereverzev. Multi-parameter regularization and its numerical realization. *Numerische Mathematik*, 118:1–31, 2011.

B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):1–10, 2018.

J. Maclean and E. S. Van Vleck. Particle filters for data assimilation based on reduced-order data models. *Quarterly Journal of the Royal Meteorological Society*, 147(736):1892–1907, 2021.

C. J. Maddison, D. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. W. Teh. Filtering variational objectives. *arXiv preprint arXiv:1705.09279*, 30, 2017.

A. J. Majda and J. Harlim. *Filtering Complex Turbulent Systems.* Cambridge University Press, 2012.

A. J. Majda and D. Qi. Strategies for reduced-order models for predicting the statistical responses and uncertainty quantification in complex turbulent dynamical systems. *SIAM Review*, 60(3):491–549, 2018.

J. Mandel, E. Bergou, and S. Gratton. 4DVAR by Ensemble Kalman Smoother. *arXiv preprint arXiv:1304.5271*, 2013.

X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *Advances in Neural Information Processing Systems*, 29, 2016.

Y. M. Marzouk and D. Xiu. A stochastic collocation approach to Bayesian inference in inverse problems. *Communications in Computational Physics*, 6(4):826–847, 2009.

Y. M. Marzouk, H. N. Najm, and L. A. Rahn. Stochastic spectral methods for efficient Bayesian solution of inverse problems. *Journal of Computational Physics*, 224(2):560–586, 2007.

R. Maulik, B. Lusch, and P. Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3):037106, 2021.

H. P. McKean. Propagation of chaos for a class of non-linear parabolic equations. *Stochastic Differential Equations (Lecture Series in Differential Equations, Session 7, Catholic Univ., 1967)*, pages 41–57, 1967.

Y. A. Melnikov and M. Y. Melnikov. Computability of series representations for Green's functions in a rectangle. *Engineering analysis with boundary elements*, 30(9):774–780, 2006.

S. Metref, A. Hannart, J. Ruiz, M. Bocquet, A. Carrassi, and M. Ghil. Estimating model evidence using ensemble-based data assimilation with localization–The model selection problem. *Quarterly Journal of the Royal Meteorological Society*, 145(721):1571–1588, 2019.

C. Naesseth, S. Linderman, R. Ranganath, and D. Blei. Variational sequential Monte Carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 968–977. PMLR, 2018.

A. S. Nemirovskij and D. B. Yudin. Problem complexity and method efficiency in optimization. *Wiley-Interscience Series in Discrete Mathematics*, 1983.

D. Nguyen, S. Ouala, L. Drumetz, and R. Fablet. Em-like learning chaotic dynamics from noisy and partial observations. *arXiv preprint arXiv:1903.10335*, 2019.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.

G. Nœvdal, T. Mannseth, and E. Vefring. Instrumented wells and near-well reservoir monitoring through ensemble kalman filter. In *8th European Conference on the Mathematics of Oil Recovery*, 2002.

E. Ott, B. R. Hunt, I. Szunyogh, A. V. Zimin, E. J. Kostelich, M. Corazza, E. Kalnay, D. Patil, and J. A. Yorke. A local ensemble Kalman filter for atmospheric data assimilation. *Tellus A: Dynamic Meteorology and Oceanography*, 56(5):415–428, 2004.

L. Palatella, A. Carrassi, and A. Trevisan. Lyapunov vectors and assimilation in the unstable subspace: theory and applications. *Journal of Physics A: Mathematical and Theoretical*, 46(25):254020, 2013.

O. Papaspiliopoulos, M. Ruggiero, et al. Optimal filtering and the dual process. *Bernoulli*, 20(4):1999–2019, 2014.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8024–8035, 2019.

B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *arXiv preprint arXiv:1806.10761*, 2018.

M. Pulido, P. Tandeo, M. Bocquet, A. Carrassi, and M. Lucini. Stochastic parameterization identification using ensemble Kalman filtering combined with maximum likelihood methods. *Tellus A: Dynamic Meteorology and Oceanography*, 70(1):1–17, 2018.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. In *Artificial intelligence and statistics*, pages 814–822. PMLR, 2014.

S. S. Rangapuram, M. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. In *Proceedings of the 32nd international conference on neural information processing systems*, pages 7796–7805, 2018.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*, volume 1. MIT press Cambridge, 2006.

S. Reich and C. Cotter. *Probabilistic Forecasting and Bayesian Data Assimilation*. Cambridge University Press, 2015.

S. Reich and C. J. Cotter. Ensemble filter techniques for intermittent data assimilation. *Large Scale Inverse Problems: Computational Methods and Applications in the Earth Sciences*, 13:91–134, 2013.

A. C. Reynolds, M. Zafari, and G. Li. Iterative forms of the ensemble Kalman filter. In *ECMOR X-10th European conference on the mathematics of oil recovery*, pages cp–23. European Association of Geoscientists & Engineers, 2006.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

C. Robert. *The Bayesian Choice: from Decision-theoretic Foundations to Computational Implementation*. Springer Science & Business Media, 2007.

M. Roth, G. Hendeby, C. Fritsche, and F. Gustafsson. The Ensemble Kalman filter: a signal processing perspective. *EURASIP Journal on Advances in Signal Processing*, 2017(1): 1–16, 2017.

Y. Rubanova, R. T. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

D. Rudolf and B. Sprungk. On a Generalization of the Preconditioned Crank–Nicolson Metropolis Algorithm. *Foundations of Computational Mathematics*, pages 1–35, 2015.

J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

P. Sakov and L. Bertino. Relation between two common localisation methods for the EnKF. *Computational Geosciences*, 15(2):225–237, 2011.

P. Sakov, D. S. Oliver, and L. Bertino. An iterative EnKF for strongly nonlinear systems. *Monthly Weather Review*, 140(6):1988–2004, 2012.

D. Sanz-Alonso. Importance sampling and necessary sample size: an information theory approach. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):867–879, 2018.

D. Sanz-Alonso and A. M. Stuart. Long-time asymptotics of the filtering distribution for partially observed chaotic dynamical systems. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1200–1220, 2015.

D. Sanz-Alonso and Z. Wang. Bayesian Update with Importance Sampling: Required Sample Size. *Entropy*, 23(1):22, 2021.

D. Sanz-Alonso, A. M. Stuart, and A. Taeb. Inverse Problems and Data Assimilation. *arXiv preprint arXiv:1810.06191*, 2018.

H. Schaeffer, G. Tran, and R. Ward. Extracting sparse high-dimensional dynamics from limited data. *SIAM Journal on Applied Mathematics*, 78(6):3279–3295, 2018.

C. Schillings and A. M. Stuart. Analysis of the ensemble Kalman filter for inverse problems. *SIAM Journal on Numerical Analysis*, 55(3):1264–1290, 2017.

C. Schillings and A. M. Stuart. Convergence analysis of ensemble Kalman inversion: the linear, noisy case. *Applicable Analysis*, 97(1):107–123, 2018.

T. Schneider, A. M. Stuart, and J.-L. Wu. Ensemble Kalman inversion for sparse learning of dynamical systems from time-averaged data. *Journal of Computational Physics*, 470: 111559, 2022.

C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ. *Analysis and Applications*, 17(01): 19–55, 2019.

B. Shi, S. S. Du, M. I. Jordan, and W. J. Su. Understanding the acceleration phenomenon via high-resolution differential equations. *Mathematical Programming*, pages 1–70, 2021.

G. Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—I. Derivation of basic equations. *Acta Astronautica*, 4(11):1177–1206, 1977.

J.-A. Skjervheim, G. Evensen, J. Hove, J. G. Vabø, et al. An ensemble smoother for assisted history matching. In *SPE Reservoir Simulation Symposium*. OnePetro, 2011.

C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640, 2008.

A. Spantini, D. Bigoni, and Y. Marzouk. Inference via low-dimensional couplings. *The Journal of Machine Learning Research*, 19(1):2639–2709, 2018.

J. R. Stroud and T. Bengtsson. Sequential state and variance estimation within the ensemble Kalman filter. *Monthly Weather Review*, 135(9):3194–3208, 2007.

J. R. Stroud, M. L. Stein, B. M. Lesht, D. J. Schwab, and D. Beletsky. An ensemble Kalman filter and smoother for satellite data assimilation. *Journal of the American Statistical Association*, 105(491):978–990, 2010.

J. R. Stroud, M. Katzfuss, and C. K. Wikle. A Bayesian adaptive ensemble Kalman filter for sequential state and parameter estimation. *Monthly Weather Review*, 146(1):373–386, 2018.

A. M. Stuart. Inverse problems: a Bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

A. M. Stuart and A. Teckentrup. Posterior consistency for Gaussian process approximations of Bayesian posterior distributions. *Mathematics of Computation*, 2017.

W. Su, S. Boyd, and E. Candes. A differential equation for modeling Nesterov's accelerated gradient method: theory and insights. *Advances in neural information processing systems*, 27, 2014.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, volume 27, pages 3104–3112, 2014.

A.-S. Sznitman. Topics in propagation of chaos. In *Ecole d'été de probabilités de Saint-Flour XIX—1989*, pages 165–251. Springer, 1991.

I. Szunyogh, E. J. Kostelich, G. Gyarmati, E. Kalnay, B. R. Hunt, E. Ott, E. Satterfield, and J. A. Yorke. A local ensemble transform Kalman filter data assimilation system for the NCEP global model. *Tellus A: Dynamic Meteorology and Oceanography*, 60(1):113–130, 2008.

P. Tandeo, M. Pulido, and F. Lott. Offline parameter estimation using EnKF and maximum likelihood error covariance estimates: Application to a subgrid-scale orography parametrization. *Quarterly journal of the royal meteorological society*, 141(687):383–395, 2015.

L. Tierney and A. Mira. Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in medicine*, 18(17-18):2507–2515, 1999.

M. K. Tippett, J. L. Anderson, C. H. Bishop, T. M. Hamill, and J. S. Whitaker. Ensemble square root filters. *Monthly Weather Review*, 131(7):1485–1490, 2003.

G. Tran and R. Ward. Exact recovery of chaotic systems from highly corrupted data. *Multiscale Modeling & Simulation*, 15(3):1108–1129, 2017.

A. Trevisan, M. D'Isidoro, and O. Talagrand. Four-dimensional variational assimilation in the unstable subspace and the optimal subspace dimension. *Quarterly Journal of the Royal Meteorological Society*, 136(647):487–496, 2010.

G. Ueno and N. Nakamura. Iterative algorithm for maximum-likelihood estimation of the observation-error covariance matrix for ensemble-based filters. *Quarterly Journal of the Royal Meteorological Society*, 140(678):295–315, 2014.

G. Ueno and N. Nakamura. Bayesian estimation of the observation-error covariance matrix in ensemble-based filters. *Quarterly Journal of the Royal Meteorological Society*, 142(698): 2055–2080, 2016.

S. Ungarala. On the iterated forms of Kalman filters using statistical linearization. *Journal of Process Control*, 22(5):935–943, 2012.

Z. Y. Wan and T. P. Sapsis. Reduced-space Gaussian Process Regression for data-driven probabilistic forecast of chaotic dynamical systems. *Physica D: Nonlinear Phenomena*, 345:40–55, 2017.

G. C. Wei and M. A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American statistical Association*, 85(411):699–704, 1990.

J. S. Whitaker, T. M. Hamill, X. Wei, Y. Song, and Z. Toth. Ensemble data assimilation with the NCEP Global Forecast System. *Monthly Weather Review*, 136(2):463–482, 2008.

A. Wibisono, A. C. Wilson, and M. I. Jordan. A variational perspective on accelerated methods in optimization. *proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016.

A. Wikner, J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, and E. Ott. Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(5):053111, 2020.

R. J. Williams and D. Zipser. Gradient-based Learning Algorithms for Recurrent Networks and their Computational Complexity. In Y. Chauvin and D. E. Rumelhart, editors, *Back-propagation: Theory, Architectures and Applications*, volume 433, chapter 13, pages 433–486. Hillsdale, NJ: Erlbaum, 1995.

M. A. Woodbury. *Inverting Modified Matrices*. Statistical Research Group, 1950.

D. Xiu and G. E. Karniadakis. The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

K. Xu and C. K. Wikle. Estimation of parameterized spatio-temporal dynamic models. *Journal of Statistical Planning and Inference*, 137(2):567–588, 2007.

L. M. Yang and I. Grooms. Machine learning techniques to construct patched analog ensembles for data assimilation. *Journal of Computational Physics*, 443:110532, 2021.

Y. Yu, X. Si, C. Hu, and J. Zhang. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, 31(7):1235–1270, 07 2019.

A. Zellner. Optimal information processing and Bayes's theorem. *The American Statistician*, 42(4):278–280, 1988.