Advanced Sampling Methods for Solving Large-Scale Inverse Problems

Ahmed Attia

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science Applications

Adrian Sandu, Chair
Calvin J. Ribbens
Jeffrey L. Anderson
Matthias Chung
Yang Cao

August 22, 2016
Blacksburg, Virginia

Keywords: Data Assimilation, Hamiltonian Monte Carlo, Parallel MCMC,
Reduced-Order Modelling, Mixture Models

Advanced Sampling Methods for Solving Large-Scale Inverse Problems

Ahmed Attia

(ABSTRACT)

Ensemble and variational techniques have gained wide popularity as the two main approaches for solving data assimilation and inverse problems. The majority of the methods in these two approaches are derived (at least implicitly) under the assumption that the underlying probability distributions are Gaussian. It is well accepted, however, that the Gaussianity assumption is too restrictive when applied to large nonlinear models, nonlinear observation operators, and large levels of uncertainty. This work develops a family of fully non-Gaussian data assimilation algorithms that work by directly sampling the posterior distribution. The sampling strategy is based on a Hybrid/Hamiltonian Monte Carlo (HMC) approach that can handle non-normal probability distributions.

The first algorithm proposed in this work is the "*HMC sampling filter*", an ensemble-based data assimilation algorithm for solving the sequential filtering problem. Unlike traditional ensemble-based filters, such as the ensemble Kalman filter and the maximum likelihood ensemble filter, the proposed sampling filter naturally accommodates non-Gaussian errors and nonlinear model dynamics, as well as nonlinear observations. To test the capabilities of the HMC sampling filter numerical experiments are carried out using the Lorenz-96 model and observation operators with different levels of nonlinearity and differentiability. The filter is also tested with shallow water model on the sphere with linear observation operator. Numerical results show that the sampling filter performs well even in highly nonlinear situations where the traditional filters diverge.

Next, the HMC sampling approach is extended to the four-dimensional case, where several observations are assimilated simultaneously, resulting in the second member of the proposed family of algorithms. The new algorithm, named "*HMC sampling smoother*", is an ensemble-based smoother for four-dimensional data assimilation that works by sampling from the posterior probability density of the solution at the initial time. The sampling smoother naturally accommodates non-Gaussian errors and nonlinear model dynamics and observation operators, and provides a full description of the posterior distribution. Numerical experiments for this algorithm are carried out using a shallow water model on the

sphere with observation operators of different levels of nonlinearity. The numerical results demonstrate the advantages of the proposed method compared to the traditional variational and ensemble-based smoothing methods.

The HMC sampling smoother, in its original formulation, is computationally expensive due to the innate requirement of running the forward and adjoint models repeatedly. The proposed family of algorithms proceeds by developing computationally efficient versions of the HMC sampling smoother based on reduced-order approximations of the underlying model dynamics. The reduced-order HMC sampling smoothers, developed as extensions to the original HMC smoother, are tested numerically using the shallow-water equations model in Cartesian coordinates. The results reveal that the reduced-order versions of the smoother are capable of accurately capturing the posterior probability density, while being significantly faster than the original full order formulation.

In the presence of nonlinear model dynamics, nonlinear observation operator, or non-Gaussian errors, the prior distribution in the sequential data assimilation framework is not analytically tractable. In the original formulation of the HMC sampling filter, the prior distribution is approximated by a Gaussian distribution whose parameters are inferred from the ensemble of forecasts. The Gaussian prior assumption in the original HMC filter is relaxed. Specifically, a clustering step is introduced after the forecast phase of the filter, and the prior density function is estimated by fitting a Gaussian Mixture Model (GMM) to the prior ensemble. The base filter developed following this strategy is named *cluster HMC sampling filter* (*C*ℓHMC). A multi-chain version of the *C*ℓHMC filter, namely MC-*C*ℓHMC , is also proposed to guarantee that samples are taken from the vicinities of all probability modes of the formulated posterior. These methodologies are tested using a quasi-geostrophic (QG) model with double-gyre wind forcing and bi-harmonic friction. Numerical results demonstrate the usefulness of using GMMs to relax the Gaussian prior assumption in the HMC filtering paradigm.

To provide a unified platform for data assimilation research, a flexible and a highly-extensible testing suite, named DATeS , is developed and described in this work. The core of DATeS is implemented in Python to enable for Object-Oriented capabilities. The main components, such as the models, the data assimilation algorithms, the linear algebra solvers, and the time discretization routines are independent of each other, such as to offer maximum flexibility to configure data assimilation studies.

Dedication

To my parents, my sister, and my brothers.

Acknowledgments

First and foremost I want to thank my advisor Prof. Adrian Sandu. He has taught me, both consciously and unconsciously, how good research life is. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating.

The members of the Computational Science Laboratory (CSL) group have contributed immensely to my personal and professional time at Virginia Tech. The group has been a source of friendships as well as good advice and collaboration.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was funded by awards NSF CCF–1218454 and AFOSR FA9550–12–1–0293–DEF, NSF DMS–1419003, AFOSR 12-2640-06, VT-MENA program, AFOSR Dynamic Data Driven Application Systems program, and by the Computational Science Laboratory (CSL) in the Department of Computer Science at Virginia Tech.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scientific context

Modern fields of science and engineering rely on sophisticated computer models to describe, predict, and understand the behavioral patterns of complex physical phenomena such as the weather, ocean dynamics, oil reservoirs, earthquakes, and volcanoes. Predictions made by a computer model form an essential source of information about the unknown true state of the physical system of concern. These predictions encapsulate the knowledge about the underlying physical phenomena "prior" to incorporating additional sources of information. In addition to the prior information characterized by the model prediction, complex sensor networks are also employed to collect measurements of the physical phenomena, producing sparse snapshots of reality. Data assimilation (DA) is the inverse problem that fuses information from priors, computer model results, and measurements of reality, in order to provide a consistent description of the true state of the system of concern[1, 2, 3].

Model description, model parameters, observations, and predictions made by DA schemes are all affected by uncertainty. A probabilistic representation of these uncertainties is incorporated in the DA process. Bayes' theorem is used to formulate the posterior distribution of the system state which fuses all information from all incorporated sources. In numerous DA applications it is common practice to assume that the errors are unbiased and follow Gaussian distributions, leading to analytically tractable solutions. While nonlinearity and non-Gaussianity are the norm in all applications of practical interest, few DA methodologies currently available can fully account for them. Two approaches are

of special interest for solving DA problems. Statistical approaches follow a Monte-Carlo strategy to approximate the posterior distribution of system state, in the Bayesian framework, based on an ensemble of model states [4, 5, 6, 7, 8, 9, 10, 11]. The variational approach emerged in control theory generally incorporates an optimization step whose solution is an *analysis* state that minimizes the mismatch between actual observations and observations predicted by the mathematical model. The analysis provided by the variational approach seeks to find the maximum a posteriori estimate of the true state. While the ensemble-based DA methodologies are widely used, most of these schemes assume that the distribution of system states is Gaussian, or at most weakly non-Gaussian. This assumption however is very restrictive in practical situations where models are complex and capable of resolving nonlinear processes, or observations are nonlinearly related to model states, or error distributions are not justifiably Gaussian. In addition to relying on the Gaussianity assumption, ensemble-based schemes tend to produce dynamically inconsistent model states. Physical imbalances can result from localization, which is usually carried out to remove spurious correlations, or from the time-discontinuous nature of the DA process. The variational schemes tend to attain lower errors, especially in the presence of infrequent observations. These methods, however, provide a single estimate of the system state, lacking a consistent description of the uncertainty associated with the provided analysis. The computational costs of the variational algorithms can be reduced by replacing the model dynamics with reduced-order approximation, by projecting the model dynamics in an affine subspace that captures most of the model physics. When the Gaussianity assumptions are severely violated, or when strong nonlinearity is evident in the model dynamics or the observation operators, the performance of these algorithms degrade, and the results might be misleading. To harness the best of the two worlds, hybrid ensemble-variational algorithms have been developed to provide analysis with less error along with a measure of the associated uncertainty [12, 13, 14, 15]. The hybrid methods in general suffer from inconsistencies mainly because the analysis and the uncertainty quantification are obtained using different strategies.

1.2 Problem formulation

Assume the true state of a physical system at a given time t_k is described by $\mathbf{x}^{\text{true}}(t_k)$. The time evolution of the physical system is approximated by the discretized forward model:

$$\mathbf{x}_{k+1} = \mathcal{M}_{k,k+1}(\mathbf{x}_k)\,, \quad k = 0, 1, \ldots, N-1\,, \tag{1.1}$$

where $\mathbf{x}_k \in \mathbb{R}^{\text{N}_{\text{state}}}$ is a discretized approximation of the true state at time instance t_k.

The prior distribution $\mathcal{P}^{\mathrm{b}}(\mathbf{x}_k)$ encapsulates the knowledge about the model state at time instance t_k before additional information is incorporated. The likelihood function $\mathcal{P}(\mathbf{Y}|\mathbf{x}_k)$ quantifies the deviation of the prediction of model observations from the collected measurements. The posterior distribution is formulated by applying Bayes' theorem as follows:

$$\mathcal{P}^{\mathrm{a}}(\mathbf{x}_k|\mathbf{Y}) = \frac{\mathcal{P}^{\mathrm{b}}(\mathbf{x}_k)\,\mathcal{P}(\mathbf{Y}|\mathbf{x}_k)}{\mathcal{P}(\mathbf{Y})}, \tag{1.2}$$

where $\mathbf{Y}$ refers to the data (observations) to be assimilated. In the sequential filtering context $\mathbf{Y}$ is a single observation, while in the smoothing context, it generally stands for several observations $\{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m\}$ to be assimilated simultaneously.

Consider assimilating information available about the system state at time instance t_k. The computer model is used to provide a prior prediction (forecast) about the system state denoted by $\mathbf{x}_k^{\mathrm{b}}$. In typical applications of DA, the error distributions are assumed to be Gaussian, resulting in the so-called "Gaussian framework". Consider a Gaussian framework, where the prior errors are assumed to be normally distributed with zero mean, and a covariance matrix $\mathbf{B}_k \in \mathbb{R}^{\mathrm{N_{state}}\times\mathrm{N_{state}}}$, that is $\left(\mathbf{x}_k^{\mathrm{b}} - \mathbf{x}^{\mathrm{true}}(t_k)\right) \sim \mathcal{N}(0, \mathbf{B}_k)$. Consider an observation $\mathbf{y}_k \in \mathbb{R}^{\mathrm{N_{obs}}}$ given at time instance t_k. The observation errors are assumed to follow a Gaussian distribution with zero mean, and covariance matrix $\mathbf{R}_k \in \mathbb{R}^{\mathrm{N_{obs}}\times\mathrm{N_{obs}}}$, that is $(\mathbf{y}_k - \mathbf{y}_k^{\mathrm{true}}) \sim \mathcal{N}(0, \mathbf{R}_k)$. Following a perfect model approach, the posterior distribution follows from 1.2 as:

$$\begin{aligned}
\mathcal{P}^{\mathrm{a}}(\mathbf{x}_k|\mathbf{y}_k) &= \frac{\mathcal{P}^{\mathrm{b}}(\mathbf{x}_k)\,\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k)}{\mathcal{P}(\mathbf{y}_k)} \\
&\propto \mathcal{P}^{\mathrm{b}}(\mathbf{x}_k)\,\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k) \\
&= \frac{(2\pi)^{-\frac{\mathrm{N_{state}}}{2}}}{\sqrt{|\mathbf{B}_k|}}\,\exp\left(-\frac{1}{2}||\mathbf{x}_k - \mathbf{x}_k^{\mathrm{b}}||^2_{\mathbf{B}_k^{-1}}\right)\frac{(2\pi)^{-\frac{\mathrm{N_{obs}}}{2}}}{\sqrt{|\mathbf{R}_k|}}\,\exp\left(-\frac{1}{2}||\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)||^2_{\mathbf{R}_k^{-1}}\right) \\
&\propto \exp\left(-\frac{1}{2}||\mathbf{x}_k - \mathbf{x}_k^{\mathrm{b}}||^2_{\mathbf{B}_k^{-1}}\right)\,\exp\left(-\frac{1}{2}||\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)||^2_{\mathbf{R}_k^{-1}}\right) \\
&= \exp\left(-\mathcal{J}(\mathbf{x}_k)\right), \\
\mathcal{J}(\mathbf{x}_k) &= \frac{1}{2}||\mathbf{x}_k - \mathbf{x}_k^{\mathrm{b}}||^2_{\mathbf{B}_k^{-1}} + \frac{1}{2}||\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)||^2_{\mathbf{R}_k^{-1}}
\end{aligned} \tag{1.3}$$

where the scaling factor $\mathcal{P}(\mathbf{y}_k)$ is dropped. $\mathcal{H}_k$ is the observation operator that maps the state space vector onto the observation space. In practical applications, the dimension of the observation space is much less than the state space dimension, that is $\mathrm{N_{obs}} \ll \mathrm{N_{state}}$.

Ensemble filtering methods such as ensemble Kalman filter (EnKF) [8, 9, 10, 11, 16, 17, 18, 19, 20, 21, 22], and maximum likelihood ensemble filter (MLEF) [23] use en-

sembles of states to represent the prior, and the posterior distribution. A prior ensemble $\mathbf{X}_k = \{\mathbf{x}(e)\}_{e=1,2,\ldots,N_{ens}}$, approximating the prior distributions, is obtained by propagating analysis states from a previous assimilation cycle at time t_{k-1} by applying 1.1. Most of the ensemble based DA methodologies work by transforming the prior ensemble into an ensemble of states collected from the posterior distribution, namely an analysis ensemble. The transformation in the EnKF framework is applied following the update equations of the well-known Kalman filter. A minimum variance estimate (MVE) of the true state of the system is obtained by averaging the analysis ensemble, while the posterior covariance is approximated by the covariance matrix of the analysis ensemble.

The maximum a posteriori (MAP) estimate of the true state is the state that maximizes the posterior probability density function (PDF). Alternatively the MAP estimate is the minimizer of the negative logarithm (negative-log) of the posterior PDF. The MAP estimate can be obtained by solving the following optimization problem:

$$\min_{\mathbf{x}_k} \mathcal{J}(\mathbf{x}_k) = \frac{1}{2}\left\|\mathbf{x}_k - \mathbf{x}_k^{\mathrm{b}}\right\|^2_{\mathbf{B}_k^{-1}} + \|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}} \ . \tag{1.4}$$

This formulates the 3-dimensional variational (3D-Var) DA problem. Unlike ensemble filtering algorithms, the optimal solution of (1.4) provides a single estimate of the true state, and does not provide a direct estimate of associated uncertainty.

Assimilating several observations $\mathbf{Y} := \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m\}$ simultaneously requires adding time as a fourth dimension to the DA problem. Let $\mathcal{P}^{\mathrm{b}}(\mathbf{x}_0)$ be the prior distribution of the system state at the beginning of a time window $[t_0, t_F]$ over which the observations are distributed. Assuming the observations errors are temporally uncorrelated, the posterior distribution of the system state at the initial time of the assimilation window t_0 follows by applying Equation (1.2) as:

$$\begin{aligned}
\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0) &\propto \mathcal{P}^{\mathrm{b}}(\mathbf{x}_0)\,\mathcal{P}(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m|\mathbf{x}_0) \\
&\propto \exp\left(-\frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}}\right\|^2_{\mathbf{B}_0^{-1}}\right) \exp\left(-\frac{1}{2}\sum_{k=0}^{m}\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}}\right) \\
&\propto \exp\Big(-\mathcal{J}(\mathbf{x}_0)\Big), \\
\mathcal{J}(\mathbf{x}_0) &= \frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}}\right\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m}\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}} \ .
\end{aligned} \tag{1.5}$$

In the statistical approach, ensemble-based smoothers such as the ensemble Kalman smoother (EnKS) are used to approximate the posterior (1.5) based on an ensemble of states. Similar to the ensemble filters, the analysis ensemble generated by a smoothing algorithm can be used to provide an MVE of the posterior first order moment. It also can be used to

provide a flow-dependent ensemble covariance matrix to approximate the posterior true second order moment.

The MAP estimate of the true state at the initial time of the assimilation window can be obtained by solving the following optimization problem:

$$\min_{\mathbf{x}_0} \mathcal{J}(\mathbf{x}_0) = \frac{1}{2} \left\| \mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}} \right\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2} \sum_{k=0}^{m} \| \mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k) \|^2_{\mathbf{R}_k^{-1}} . \tag{1.6}$$

This is the standard formulation of the strong-constraint four-dimensional variational (4D-Var) DA problem. The solution of the 4D-Var problem is equivalent to the MAP of the smoothing posterior in the Gaussian framework. Similar to the 3D-Var case, the solution of Equation 1.6 provides a single best estimate (the analysis) of the system state without providing consistent description of the uncertainty associated with this estimate.

In idealized settings, where the model is linear, the observation operator is linear, and the underlying probability distributions are Gaussian, the posterior is also Gaussian, however this is rarely the case in real applications. Algorithms capable of accommodating non-Gaussianity are too limited and have not been successfully tested in large-scale settings. This dissertation develops a family of algorithms capable of accommodating non-Gaussian distributions as well as nonlinear model dynamics and nonlinear observation operators. The presented family of algorithms consists of both filters and smoothers those work by directly sampling from the posterior distribution following an accelerated Monte-Carlo approach.

1.3 Research objectives

The overall goal of this dissertation is to advance the state-of-the-science in the solution procedures and in the uncertainty quantification methodologies for data assimilation problems. Specifically, our research objectives are to:

1. develop novel methodologies for performing DA with highly-nonlinear models and in the presence of non-Gaussian probabilistic uncertainties, and to
2. develop and implement a unified and highly-extensible testing suite for data assimilation research, following an object-oriented programming approach.

Research objective 1. Ensemble-based data assimilation algorithms such as the Ensemble Kalman filter (EnKF) have gained wide popularity due to their simple formulation,

ease of implementation, and good practical results. Variational schemes, such as the four-dimensional variational (4D-Var) algorithm, provide a single best estimate of the uncertain state of the physical system of concern. Methodologies in the variational approach usually result in lower errors than ensemble-based algorithms, especially in the presence of infrequent observations. The variational algorithms, however, are very expensive as they require several evaluations of the forward, and the backward (e.g. adjoint) model, during the optimization process. These algorithms also do not provide a consistent quantification of the uncertainty associated with the resulting optimal state, e.g. the "analysis". *Our objective is to develop a family of efficient filtering and smoothing algorithms capable of accommodating non-Gaussian probability distributions as well as non-linearity in the model dynamics and the observation operators.* A consistent and efficient family of algorithms for solving non-Gaussian large-scale data assimilation problems, can potentially increase the accuracy of the forecasts carried out operational centers including weather forecast centers.

Research objective 2. The data assimilation research process requires carrying out numerous numerical experiments in standardized settings to asses the usefulness of a proposed idea or algorithm. The available data assimilation testing suites are either tied to specific settings, or are very general and require a considerable learning overhead. Moreover, existing testing suites for data assimilation applications are usually completely written in a specific language. *Our objective is to develop a highly-extensible unified data assimilation testing platform capable of interfacing with third party codes written in different languages.* An open-source central testing suite would enable researchers to add different pieces of the data assimilation process and test them against existing methodologies with minimal effort. Having such an extensible testing suite can potentially aid in the improvement of the data assimilation methodologies.

1.4 Current state-of-the-art

We now discuss the cutting-edge methodologies available in the literature that are relevant in the context of our research objectives.

1.4.1 Non-Gaussian data assimilation algorithms

Most of the operational DA problems are in fact nonlinear and non-Gaussian. Nonlinearity can come from nonlinear forward (observation) operator, nonlinear model dynamics or both. As a result of nonlinearity, the probability distributions involved in the DA formulation are non-Gaussian. DA methodologies developed to accommodate non-Gaussianity have been successfully applied only in very low-dimensional systems. As a result, the linear (or linearized) formulation of the DA problem has been relied on heavily in the literature. While the linear/linearized approach has been operationally adopted for practical purposes, the real systems are highly nonlinear.

Zupanski [23] developed the maximum likelihood ensemble filter (MLEF) to deal with nonlinear observations by posing a nonlinear estimation problem in a subspace spanned by the ensemble members, and to compute the maximum a posteriori estimate in that subspace. Sakov et al. [17] presented an iterative ensemble Kalman filter (IEnKF) as an extension to MLEF to handle nonlinearity in observations and models by repeating analyses steps with the same observations. Anamorphosis function [24] provides a means to incorporate a non-linear change of variables to extend EnKF to nonlinear settings by executing the analysis step in a space where the distributions of the transformed variables are Gaussian. These methodologies however do not provide a full description of the posterior distribution of the system state, especially when the posterior is multimodal.

Another approach to handle non-Gaussianity, in the EnKF context, is to incorporate Gaussian mixture models (GMM) as means of density estimation. In [25, 22], the authors incorporate a GMM to approximate the prior density based on an ensemble of forecasts. In [25], given a Gaussian likelihood function, the posterior is formulated as a GMM with updated mixture parameters. The updated parameters are obtained by executing a convolution of Gaussian distributions. The limitation with this approach however, is that the dispersion of the mixture components is unknown, and has to be tuned by trail and error. Smith [22] approaches this issue by fitting a GMM to the prior ensemble using Expectation Maximization (EM) algorithm with number of mixture components detected using widely used model selection criterion, the Akaike information criterion (AIC). The EnKF equations are then applied to each of the components in the mixture distribution to generate an analysis ensemble from the GMM posterior. These approaches, however, are limited to Gaussian likelihood functions, and posterior distribution that can be formulated as a Gaussian mixture distribution.

Particle filters (PF) [26, 27, 28, 29] are an attractive family of nonlinear and non-Gaussian methods. This family of filters is known to suffer from filtering degeneracy, especially

in large-scale systems. While PFs make no assumptions on the shape of the underlying probability distribution functions, they are not generally efficient without expensive tuning. Ades and Van-Leeuwen described the equivalent-weights PF [30, 31, 32] as an efficient enhancement of the standard PF formulation. This filter works by shifting the particles towards observations and consequently towards high-probability regions of the posterior PDF by incorporating a relaxation proposal density. It tries to avoid filter degeneracy by incorporating an equivalent-weights proposal density to insure that the retained particles are given similar (equivalent) relative weights. Tuning these proposals however is a limitation of this filter. Moreover, the samples generated by this filter are not generally guaranteed to be collected from the target posterior.

While the weak-constraint four-dimensional variational (4D-Var) formulation handles non-Gaussianity surprisingly well in practice, it does not provide consistent description of the uncertainty of the solution [33]. To the best of our knowledge, statistically consistent non-Gaussian smoothing algorithms are yet to be developed.

1.4.2 Data assimilation testing platforms

Numerical experiments play a central role in the assessment of the quality of a new idea or algorithm in the DA context.

To maximize usability of a DA testing platform, it should allow existing pieces of code to be easily "*inherited*". The idea of designing a general-purpose DA testing suite following an object-oriented approach has been around for several years, however the success stories are limited.

A closely related approach, modular-programming, is adopted by the DA Research Section (DAReS) at the National Center for Atmospheric Research (NCAR). The Data Assimilation Research Testbed (DART) [34] developed and maintained by DAReS is a DA testing pad designed to serve as a community facility for ensemble filtering. DART is the most complex and widely used DA platform to date, and is de facto standard tool used by the community. DART can be mainly used to test different existing, and new flavors, of the ensemble Kalman filtering algorithm with low-order and operational models. DART is mainly written in Fortran to achieve the desired levels of efficiency, and to enable for parallelization. DART has a long history of development and maintenance, and is currently interfaced with many practical geophysical models. Moreover it gives access to practical, and well-established parallel algorithms. While DART supports reusability by following a modular programming approach, an object-oriented approach would enable for higher levels of reusability.

The Parallel Data Assimilation Framework (PDAF) [35] is another software environment for parallel ensemble-based DA developed and maintained by computing center of the Alfred Wegener Institute. The standardized interface provided by PDAF separates the numerical model from the assimilation routines.

To the best of our knowledge, there is no object-oriented-based, general-purpose, DA testing suite that allows elements of the DA problem to be easily incorporated in both ensemble-based and variational contexts.

1.5 Research accomplishments

This section summarizes the main contributions of this dissertation.

1.5.1 A non-Gaussian sampling filter.

Following a Hamiltonian Monte-Carlo (HMC) approach for sampling, in [36] I have developed a fully non-Gaussian DA filter (HMC sampling filter) capable of handling non-Gaussianity by sampling directly from the posterior distribution. The forecast step is carried out by propagating the analysis ensemble members, using model dynamics, to the current assimilation time. In the analysis step, the HMC sampler is used to generate an ensemble of states collected efficiently from the posterior distribution. We show that the choice and tuning of the symplectic integrator, used to propagate the auxiliary Hamiltonian system, play central roles in the robustness of the sampler, particularly in highly nonlinear settings. The filter is capable of navigating the state space regions with high posterior probability, and provides correct statistics describing the analysis and the associated uncertainties.

1.5.2 A non-Gaussian sampling smoother.

Smoothing is necessary in applications such as hydrology and meteorology. To meet the needs of these applications, I have extended the filter design to assimilate several observations simultaneously by designing a non-Gaussian HMC smoother [37, 38]. Unlike 4D-Var DA, the HMC sampling smoother provides a consistent description of the posterior distribution of the model state at the assimilation time instance. The smoother provides

an ensemble of model states collected directly from the posterior distribution of the initial state at the beginning of the assimilation window.

1.5.3 Efficient versions of the HMC smoother using reduced-order approximations.

The information consistency provided by the HMC smoother comes at the expense of higher computational cost compared with 4D-Var. As a first attempt to develop practical versions of the HMC sampling smoother, we proposed replacing the expensive terms and steps in the smoother with reduced-order alternatives [39]. Specifically, in order to reduce the computational time of the HMC smoother, we project the dynamics onto the reduced-order subspaces obtained by methods like proper orthogonal decomposition (POD). Our results indicate that the reduced-order HMC smoothers can provide a high quality approximation of the results obtained by the original HMC smoother while being significantly faster.

1.5.4 Relaxing the Gaussian-prior assumption in the HMC sampling filter.

In the previous work, I have formulated the posterior distribution assuming the prior distribution is Gaussian. This assumption was not mandatory but was made for simplicity since tracking the posterior distribution exactly after several cycles is not possible in the DA context. This is mainly a result of propagating non-Gaussian posteriors using highly-nonlinear model dynamics. Gaussian mixture models (GMM) can potentially approximate any probability distribution. We have developed a fully non-Gaussian version of the HMC sampling filter, named Cluster HMC sampling ($C\ell$HMC)filter, that incorporates a GMM to relax the Gaussian-prior assumption posed in the original formulation. A more efficient version, multi-chain $C\ell$HMC (MC-$C\ell$HMC) filter is also introduced to guarantee sampling from the vicinities of all probability modes of the posterior. With stronger nonlinearity e.g. in the model or the observation operator, the developed versions outperform the original HMC sampling filter.

1.5.5 DATeS.

An essential component of the data assimilation research process is the use of a testing platform to investigate the usefulness of a newly proposed idea.

A flexible and a highly-extensible testing suite, named DATeS , is developed and described in the work. DATeS is aimed to be a unified testing suite for data assimilation applications where researchers can collaborate, so that it would be much easier to understand and compare different methodologies in different settings. The core of DATeS is implemented in Python to enable for Object-Oriented capabilities. The main functionalities, such as the models, the data assimilation algorithms, the linear algebra solvers, and the time discretization routines are independent of each other, such as to offer maximum flexibility to configure data assimilation applications. DATeS can interface easily with large third party models written in Fortran or C, and with various external solvers. The family of algorithms presented in this work is implemented in DATeS along with standard versions of the traditional data assimilation algorithms.

1.6 Dissertation layout

The rest of this dissertation details the research accomplishments described in the previous section. The dissertation is organized as follows:

- Chapter 2 develops a non-Gaussian data assimilation filtering algorithm (HMC sampling filter) that works by sampling the posterior following a Hybrid Monte-Carlo sampling approach. The formulation of the algorithm assumes the prior distribution can be approximated using a Gaussian density for simplicity. The performance of algorithm is discussed with several models of different scales, and in the presence of observation operators of various levels of non-linearity.

- Chapter 3 extends the design of the HMC sampling filter to the four dimensional case. The performance of the developed algorithm (HMC sampling smoother) with small, and relatively large-scale models in various settings is discussed.

- Chapter 4 develops efficient versions of the HMC sampling smoother by introducing reduced-order approximation of the model dynamics. Numerical experiments, to test the performance of these algorithms, are carried out using a shallow-water equations (SWE) model in Cartesian coordinates.

- In Chapter 5, we relax the Gaussian-prior assumption posed in the original formulation of the HMC sampling filter. Specifically, we introduce a clustering step in the forecast phase of the filter to build more accurate density approximation of the prior distribution. Performance of the developed algorithms ($C\ell$HMC , and MC-$C\ell$HMC) is tested using a quasi-geostrophic (QG) model with double-gyre wind forcing and bi-harmonic friction.
- In Chapter 6, we describe DATeS ; a unified and highly-extensible data assimilation testing suite.
- Chapter 7 draws the conclusions and points to future research directions.

Chapter 2

A Hybrid Monte Carlo Sampling Filter for Non-Gaussian Data Assimilation

2.1 Introduction

Data assimilation (DA) is the process of combining information from models, measurements, and priors - all with associated uncertainties - in order to better describe the the true state of a physical system. The ultimate goal of DA is to efficiently represent the posterior PDF that fully describes the uncertainty in the analysis. Two families of methods, variational and ensemble based methods, have proved very successful in real applications. Variational methods, rooted in control theory, require costly developments of tangent linear and adjoint models [1]. Ensemble-based sequential DA schemes are rooted in statistical estimation theory. The ensemble Kalman Filter was introduced by Evensen [20] and has undergone considerable developments since then. EnKF formulations fall in one of two classes, namely stochastic or deterministic formulations [8]. In the stochastic approach, each ensemble member is updated using a perturbed version of the observation vector [10, 11]. In the deterministic formulation (which leads to square root ensemble filters [5, 6, 7, 8, 9] no observation noise is added, but transformations of the covariance matrix are applied such as to recover the correct analysis statistics.

All variants of the EnKF work well in case of linear observations [4], however in real applications the observation operators are in general nonlinear. EnKF can accommodate nonlinear observation operators using linearization, in the spirit of the extended Kalman filter [23]. An alternative approach to handle the non-linearity of observation operators is

to use the difference between nonlinear operators evaluated at two states instead of the linearized version; this approach can result in mathematical inconsistencies [23]. A different approach to deal with nonlinear observations is to pose a nonlinear estimation problem in a subspace spanned by the ensemble members, and to compute the maximum a posteriori estimate in that subspace. This leads to the maximum likelihood ensemble filter (MLEF) proposed by Zupanski [23]. MLEF minimizes a cost function that depends on nonlinear observation operators. MLEF doesn't require the observation operator to be differentiable and uses a difference approximation of the Jacobian of the observation operator. However, this approach may diverge if the observation operator is highly nonlinear or the observations are very sparse. In addition it is inherently assumed that the posterior distribution is Gaussian; the MLEF maximum a posteriori probability estimate may face difficulties in case of multimodal distributions. The iterative ensemble Kalman filter (IEnKF) [40, 17] handles nonlinearity in observations and models by repeating analyses steps with the same observations. Each iteration of IEnKF still assumes that the underlying probability distributions are Gaussian and the analysis state is the mode of the posterior. The "running in place" (RIP) EnKF scheme [41] uses a no-cost EnKS and repeatedly assimilates the observations over each assimilation window several times. Another nonlinear extension of EnKF incorporates a non-linear change of variables (anamorphosis function) [24] and executes the analysis step in a space where the distributions of the transformed variables are Gaussian.

Particle filters(PF) [26, 27, 28, 29] are another family of nonlinear and non-Gaussian methods. PF performs well with small dimensional systems. For large dimensional systems, or when the number of independent observations increases, PF is more likely to suffer from filter degeneracy [42], a situation in which only one particle gets the significant weight and the all the other weights become close to zero. Development of efficient particle filters for large scale nonlinear and non-Gaussian DA is an active area of research. For example, in the merging particle filter (MPF) [43], the posterior particles are formulated as linear combinations of the resampled particles from the prior at the measurement time, in an attempt to reduce the variance in the weights and to avoid filter degeneracy. Although the mean and the covariance of the posterior are preserved, this filter does not guarantee the shape of the posterior PDF is preserved in the filtered ensemble. As mentioned in [43] if the posterior PDF is significantly non-Gaussian, the MPF will likely produce rather poor estimates. The performance of EnKF, MLEF, and PF when observation operators are highly nonlinear was tested in [44, 45].

A promising approach for sequential Monte Carlo sampling from the posterior density is the implicit particle filter [46]. This algorithm directs the sampling towards the regions of high density areas in the posterior and in this way it controls the number of particles

in case of very high dimensional state spaces. The implicit sampling filter, however, is expensive: it requires solving a set of large algebraic equations for each particle. Another enhancement over the standard particle filter is the equivalent-weights particle filter [30, 31, 32] which incorporates a relaxation proposal density to steer the particles towards observations and consequently towards high-probability regions of the posterior PDF, and also uses an equivalent-weights proposal density to insure that the retained particles are given similar relative weights thus avoiding filter degeneracy.

In addition to the need of fully non-Gaussian DA schemes, efficient ensemble replenishment methods are needed to support parallel implementations of the current ensemble-based DA algorithms. In parallel implementations of EnKF each ensemble member typically runs on a separate set of nodes, and when any of the nodes dies out the corresponding ensemble member is lost. The ensemble number of members is reduced and after several node failures the entire EnKF DA process is in jeopardy. To continue the EnKF process efficiently, it is necessary to replace the lost ensemble member with a new one that is drawn independently from the underlying probability distribution.

Markov Chain Monte Carlo (MCMC) is a family of algorithms designed to sample from a given probability distribution by generating a Markov chain whose invariant (stationary) distribution is the target PDF. The traditional wisdom considers MCMC sampling methods to be a gold standard [47] in DA, which is however impractical in case of high dimensions due the massive computational cost required to achieve convergence and explore the whole state space. Development of scalable MCMC algorithms to sample efficiently from high dimensional space is an active research area.

A very promising approach for large systems is Hybrid Monte Carlo (HMC), a variant of MCMC sampling that incorporates an auxiliary variable and takes advantage of the properties of Hamiltonian system dynamics [48]. HMC has been considered before in the context of DA. [49] solve a nonlinear inverse problem that finds a solution to the shallow water equations such as to minimize the residual between the trace of this solution and ill-posed boundary conditions. Once the minimum is reached posterior error statistics are approximated by sampling the nearby states using HMC. They use a simulated annealing framework, where at low temperatures a minimum is obtained, and at high temperatures a posterior sample is obtained. [4, Chapter 6] discusses the solution of weak-constraint 4D-Var problem using a gradient method and employing HMC to estimate the analysis errors. The use of simulated annealing with HMC random walk as an alternative to gradient-based minimization is discussed in [49, 4].

Alexander, Eiyink, and Restrepo [50] advocate the use of HMC for solving smoothing problems. They employ a generalized version of HMC aimed at shortening the decor-

relation, i.e., the number of chain steps needed to ensure independence of the generated states. Generalized HMC uses the dynamics of a modified system that is no longer Hamiltonian. There are several differences between this work and [50]. In [50] the model error, represented by additive random noise, is the only source of uncertainty; here we consider the state of the system to be uncertain due to uncertainties in initial conditions and model equations. A leap-frog discretization is employed in [50], while here we consider high order symplectic time discretizations. Finally, they test their method on a one-dimensional system. We note that the generalized HMC method employed in [50] uses the dynamics of a modified system that is no longer Hamiltonian. Finally, the tests performed in [50] focus on a one-dimensional system, while here we carry out the numerical experiments with a multidimensional nonlinear model.

The current advances in sampling algorithms make it feasible to solve inverse problems with large models by directly sampling from the posterior probability distribution. This work develops a sequential ensemble-based DA filtering technique that can accommodate non-Gaussian posterior distributions and can be efficiently applied in operational situations. Moreover, it can be used as an ensemble replenishment tool to support parallel implementations of other ensemble based DA schemes. The approach is based on applying HMC sampling in a recursive manner to solve the sequential filtering problem. The new fully nonlinear sampling filter can accommodate nonlinear observation operators and it does not require the target probability distribution to be Gaussian. The filter is open to further improvements such as the use of the generalized (nonlocal) version described in [50]. We note that a smoothing approach using MCMC to sample the posterior distribution is discussed in [51, 47], however that approach is not based on HMC.

The chapter is organized as follows. An overview of the DA problem and widely-used solution strategies is given in Section 2.2. Sampling MCMC and HMC algorithms are summarized in Section 2.3. The proposed sampling filter is presented in Section 2.4. Numerical experiments, and a comparison of the sampling filter against the traditional EnKF, MLEF, and IEnKF methods, are given in Section 2.5. Conclusions are drawn in Section 2.6.

2.2 Data Assimilation

This section provides a brief overview of the DA problem and of several solution strategies, and highlights the motivation behind the present research.

2.2.1 Problem formulation

DA combines information from prior (background) knowledge, a numerical model, and observations, all with associated errors, to describe the posterior distribution of the state of a physical system.

The background represents the best estimate of the true state *prior* to any measurement being available. The background errors (uncertainties) are typically assumed to have a Gaussian distribution $\mathbf{x}^{\mathrm{b}} - \mathbf{x}^{\mathrm{true}} \in \mathcal{N}(0, \mathbf{B})$, where $\mathbf{x}^{\mathrm{b}}$ is the background state, $\mathbf{x}^{\mathrm{true}}$ is the unknown true state, and $\mathbf{B}$ is the background error covariance matrix. The Gaussian assumption is widely used and we will follow it as well. As discussed in Section 2.4 this assumption can be inaccurate, due to the recursive application of the sampling filter, and better approximations to the prior distribution might be beneficial.

The numerical model propagates the initial model state (initial condition) $\mathbf{x}_0 \in \mathbb{R}^{\mathrm{N_{state}}}$ at time t_0 to future states $\mathbf{x}_k \in \mathbb{R}^{\mathrm{N_{state}}}$ at times t_k:

$$\mathbf{x}_k = \mathcal{M}_{t_0 \to t_k}(\mathbf{x}_0)\ , \quad t_0 \leq t_k \leq t_F\ , \tag{2.1}$$

where t_0 and t_F are the beginning and the end points of the simulation time interval. The model solution operator $\mathcal{M}$ represents, for example, a discrete approximation of the partial differential equations that govern the evolution of the dynamical system (e.g., the atmosphere). The state space is typically large, e.g., $\mathrm{N_{state}} \sim 10^6 - 10^9$ variables for atmospheric simulations.

Small perturbations $\delta\mathbf{x}$ of the state of the system evolve according to the tangent linear model:

$$\delta\mathbf{x}_k = \mathbf{M}_{t_0 \to t_k}(\mathbf{x}_0) \cdot \delta\mathbf{x}_0\ , \quad t_0 \leq t_k \leq t_F, \tag{2.2}$$

where $\mathbf{M} = \mathcal{M}'$ is the linearized model solution operator.

Observations of the true state are available at discrete time instants t_k, $t_0 \leq t_k \leq t_F$,

$$\mathbf{y}_k = \mathbf{y}(t_k) = \mathcal{H}_k(\mathbf{x}_k) + \varepsilon_k, \quad k = 0, 1, \ldots, m-1.$$

The observation operator $\mathcal{H}_k$ maps the state space to the observation space at time t_k. The observations are corrupted by measurement and representativeness errors [52], which are also assumed to have a normal distribution, $\varepsilon_k \in \mathcal{N}(0, \mathbf{R}_k)$, where $\mathbf{R}_k$ is the observation error covariance matrix at time t_k.

Data assimilation combines the background estimate, the measurements , and the model to obtain an improved estimate $\mathbf{x}^{\mathrm{a}}$, called the "*analysis*" (or *posterior*), of the true state

$\mathbf{x}^{\text{true}}$, together with the corresponding analysis probability distribution. Two approaches for solving the DA problem have gained widespread popularity, variational and ensemble-based methods. The sampling filter proposed in this chapter belongs to the latter family.

Since EnKF variations are the most popular ensemble-based algorithms in practice, we compare the new methodology against EnKF, MLEF, IEnKF. These schemes are briefly reviewed in the following subsections.

2.2.2 The ensemble Kalman filter

Kalman filters (KF) [53, 54] are sequential DA methodologies, where measurements are incorporated at the time moment when they become available. Sequential DA algorithms proceed in two steps, namely, *forecast* and *analysis*. In the forecast step, the state of the system is propagated forward by the model equations (2.1) to the next time point where observations are available, producing a forecast of the state of the system, and a forecast error covariance matrix is presented to quantify the uncertainty of the forecast.

The ensemble Kalman filter (EnKF) [10, 20, 18, 11] takes a Monte-Carlo approach to representing the uncertainty. An ensemble of N_{ens} states ($\mathbf{x}^{\text{a}}_{k-1}(e)$, $e = 1, \ldots, \mathrm{N}_{\text{ens}}$) is used to sample the analysis probability distribution at time t_{k-1}. Each member of the ensemble is propagated to t_k using the nonlinear model (2.1) to obtain the "forecast" ensemble

$$\mathbf{x}^{\text{f}}_k(e) = \mathcal{M}_{t_{k-1}\rightarrow t_k}(\mathbf{x}^{\text{a}}_{k-1}(e)) + \eta_k(e), \quad e = 1, \ldots, \mathrm{N}_{\text{ens}}. \tag{2.3a}$$

To simulate the fact that the model is an imperfect representation of reality model errors are added. They are typically considered Gaussian random variables, $\eta_k \in \mathcal{N}(0, \mathbf{Q}_k)$. The ensemble mean and covariance approximate the background estimate and the background error covariance of the state at the next time point t_k:

$$\overline{\mathbf{x}}^{\text{f}}_k = \frac{1}{\mathrm{N}_{\text{ens}}} \sum_{e=1}^{\mathrm{N}_{\text{ens}}} \mathbf{x}^{\text{f}}_k(e)\,, \tag{2.3b}$$

$$\mathbf{X}^{\text{f}}_k = [\mathbf{x}^{\text{f}}_k(1) - \overline{\mathbf{x}}^{\text{f}}_k, \ldots, \mathbf{x}^{\text{f}}_k(\mathrm{N}_{\text{ens}}) - \overline{\mathbf{x}}^{\text{f}}_k]\,, \tag{2.3c}$$

$$\mathbf{B}_k = \left(\frac{1}{\mathrm{N}_{\text{ens}} - 1} \left(\mathbf{X}^{\text{f}}_k \left(\mathbf{X}^{\text{f}}_k \right)^T \right) \right) \circ \rho. \tag{2.3d}$$

To reduce sampling error due to the small ensemble size, localization [19, 21, 9] is performed by taking the point-wise product of the ensemble covariance and a decorrelation matrix ρ.

Each member of the forecast (ensemble of forecast states $\{\mathbf{x}_k^{\mathrm{f}}(e)\}_{e=1,\dots,\mathrm{N_{ens}}}$) is analyzed separately using the KF formulas [10, 20]

$$\mathbf{x}_k^{\mathrm{a}}(e) = \mathbf{x}_k^{\mathrm{f}}(e) + \mathbf{K}_k\left(\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k^{\mathrm{f}}(e)) + \zeta_k(e)\right), \tag{2.4a}$$

$$\mathbf{K}_k = \mathbf{B}_k\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{B}_k\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}. \tag{2.4b}$$

The stochastic ("perturbed observations") version [10] of the ensemble Kalman filter adds a different realization of the observation noise $\zeta_k \in \mathcal{N}(0, \mathbf{R}_k)$ to each individual assimilation. The Kalman gain matrix $\mathbf{K}_k$ makes use of the linearized observation operator $\mathbf{H}_k = \mathcal{H}_k'(\overline{\mathbf{x}}_k^{\mathrm{f}})$.

Square root versions (deterministic formulations) of EnKF [8] avoid adding random noise to observations, and thus avoid additional sampling errors. The mean of the forecast ensemble is updated to produce the analysis state (posterior mean), and the posterior ensemble is induced after updating a matrix of state deviations from the ensemble mean.

The main limitation of the EnKF is due to the Gaussianity assumption on which the Kalman updates are based. The filter is optimal only when both the forecast and the observation errors are Gaussian, and the observations are linearly related to system state.

2.2.3 The maximum likelihood ensemble filter

The maximum likelihood ensemble filter (MLEF) [23] seeks to alleviate the limitations of the Gaussian assumptions by computing the maximum likelihood estimate of the state in the ensemble space. Specifically, it maximizes the posterior probability density, or equivalently, minimizes the following nonlinear objective function over the ensemble subspace [55, 23]:

$$\mathbf{x}_k^{\mathrm{opt}} = \arg\min_{\mathbf{x}} \mathcal{J}(\mathbf{x}), \tag{2.5a}$$

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{x}_k^{\mathrm{b}}\|^2_{\mathbf{B}_k^{-1}} + \frac{1}{2}\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x})\|^2_{\mathbf{R}_k^{-1}}, \tag{2.5b}$$

and then updates the analysis error covariance matrix based on the fact that it is approximately equal to the inverse of the Hessian matrix at the minimum [56]. An important advantage of the algorithm is that the observation operator is partly linearized only in the ensemble perturbations. Consequently MLEF can work efficiently with non-linear observation operators (without the requirement of differentiability and without using finite-difference approximations of the Jacobian of the observation operators) [23].) The cost function (2.5b) to minimize implicitly assumes that the posterior distribution is Gaussian.

The method is unlikely to give good results when the posterior distributions are multimodal.

2.2.4 The iterative ensemble Kalman filter

As discussed in Section 2.2.3 MLEF follows an iterative minimization procedure of a penalty (cost) functional in the subspace spanned by the ensemble members. MLEF is specifically designed to handle nonlinear observations but may fail in the presence of strong nonlinearity in the governing model [17]. The ensemble iterative randomized maximum likelihood (EnRML) filter [40], and the iterative ensemble Kalman filter (IEnKF) [17] are both developed as extensions of MLEF. Following an iterative minimization of the cost functional with EnKF as linear solution, both EnRML and IEnKF aim at handling highly nonlinear models as well as nonlinear observation operators. A deterministic approach to the EnKF is used as base for IEnKF, while the stochastic (perturbed observations) approach is followed in the case of EnRML. IEnKF also rescales the ensemble anomalies with the ensemble transform matrix from the previous iteration, rather than using a linear regression to estimate the sensitivities between the ensemble observations and ensemble anomalies [17]. Although IEnKF is very powerful in handling nonlinearity, it is not expected to perform well if the posterior is multimodal. The optimal solution obtained can be a local minimum of the cost functional resulting in an inaccurate representation of the true state of the system. Details of IEnKF with a simple, yet detailed, pseudo code can be found in [17]. We use the formulation of IEnKF presented in [17] to check the validity of the analysis produced by the sampling filter in the cases where nonlinear observation operators are used. The IEnKF parameters (inflation factor and tolerance) are carefully tuned in order to give the best possible analysis state with minimum RMSE. This requires more iterations of the IEnKF optimization step, but it will also allow us to check how close sampling filter estimate can approach the optimal solution.

2.3 Hybrid Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) algorithms [57], introduced by Metropolis *et. al* [58], can sample from distributions with complex probability densities $\pi(\mathbf{x})$. They generate a Markov chain $\{\mathbf{x}(i)\}_{i\geq 0}$ for which $\pi(\mathbf{x})$ is the invariant (stationary) distribution, given that $\pi(\mathbf{x})$ is known up to a multiplicative constant [57]. MCMC methods work by generating a random walk using a proposal PDF and an "*acceptance/rejection*" criterion to decide whether proposed samples should be accepted as part of the Markov chain or should just

be rejected. These algorithms are generally powerful, but may take a long time to explore the whole state space or even to converge [59]. This section starts with a review of the Hamiltonian Monte-Carlo sampling (HMC) then presents the sampling filter algorithm for DA.

Hybrid Monte Carlo (HMC) methods, also known as Hamiltonian Monte Carlo, originated in the physics literature [48]. They attempt to handle the drawbacks of MCMC algorithms by incorporating an auxiliary variable such as to reduce the correlation between successive samples, to explore the entire space in very few steps, and to ensure high probability of acceptance for proposed samples in high dimensions [57].

2.3.1 Hamiltonian dynamics

Hamiltonian dynamical systems operate in a phase space of points $(\mathbf{p}, \mathbf{x}) \in \mathbb{R}^{2\mathrm{N_{state}}}$, where the individual variables are the position $\mathbf{x} \in \mathbb{R}^{\mathrm{N_{state}}}$ and the momentum $\mathbf{p} \in \mathbb{R}^{\mathrm{N_{state}}}$. The total energy of the system is described by the Hamiltonian function $H(\mathbf{p}, \mathbf{x})$. The dynamics of the system in time is described by the following ordinary differential equations:

$$\frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{p}} H\,, \qquad \frac{d\mathbf{p}}{dt} = -\nabla_{\mathbf{x}} H. \tag{2.6}$$

The time evolution of the system (2.6) in state space is described by the flow [60, 61]

$$\mathcal{S}_T : \mathbb{R}^{2\mathrm{N_{state}}} \to \mathbb{R}^{2\mathrm{N_{state}}}; \quad \mathcal{S}_T\big(\mathbf{p}(0), \mathbf{x}(0)\big) = \big(\mathbf{p}(T), \mathbf{x}(T)\big)\,, \tag{2.7}$$

which maps the initial state of the system $(\mathbf{p}(0), \mathbf{x}(0))$ to $(\mathbf{p}(T), \mathbf{x}(T))$, the state of the system at time T.

In practical computations the analytic flow $\mathcal{S}_T$ is replaced by a numerical solution using a time reversible and symplectic numerical integration method [61, 62]. In this chapter we use five different high order symplectic integrators based on Strang's splitting formula: Verlet (Störmer, Leapfrog) algorithm (A.2) [50, 57], higher order integrators namely, two-stage (A.3), three-stage (A.4), and four-stage (A.5) position splitting integrators from [63], and the Hilbert space integrator (A.6) from [64]. The methods are summarized in A.2. To approximate $\mathcal{S}_T$ the integrator at hand takes m steps of size $h = T/m$. With a slight abuse of notation we will also denote by $\mathcal{S}_T$ the flow of the numerical solution.

2.3.2 HMC sampling algorithm

In order to draw samples $\{\mathbf{x}(e)\}_{e\geq 0}$ from a given probability distribution $\pi(\mathbf{x})$ HMC makes the following analogy with a Hamiltonian mechanical system (2.6). The state $\mathbf{x}$ is viewed as a "position variable", and an auxiliary "momentum variable" $\mathbf{p}$ is included. The Hamiltonian function of the system is:

$$\begin{aligned} H(\mathbf{p},\mathbf{x}) &= \frac{1}{2}\,\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} - \log(\pi(\mathbf{x})) \\ &= \frac{1}{2}\,\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} + \mathcal{J}(\mathbf{x})\,. \end{aligned} \tag{2.8}$$

The negative logarithm of the target probability density $\mathcal{J}(\mathbf{x}) = -\log(\pi(\mathbf{x}))$ is viewed as the potential energy of the system. The kinetic energy of the system is given by the auxiliary momentum variable $\mathbf{p}$. The constant positive definite symmetric "mass matrix" $\mathbf{M}$ is yet to be defined. Based on the Hamiltonian equations (2.6) the dynamics of the system is given by

$$\frac{d\mathbf{x}}{dt} = \mathbf{M}^{-1}\mathbf{p}\,, \qquad \frac{d\mathbf{p}}{dt} = -\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}). \tag{2.9}$$

The canonical probability distribution of the state of the system $(\mathbf{p},\mathbf{x})$ in the phase space $\mathbb{R}^{2\mathrm{N_{state}}}$ is, up to a constant, equal to

$$\begin{aligned} \exp\left(-H(\mathbf{p},\mathbf{x})\right) &= \exp\left(-\frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} - \mathcal{J}(\mathbf{x})\right) \\ &= \exp\left(-\frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}\right)\cdot\pi(\mathbf{x})\,. \end{aligned} \tag{2.10}$$

The product form of this joint probability distribution shows that the two variables $\mathbf{p},\mathbf{x}$ are independent. The marginal distribution of the momentum variable is Gaussian, $\mathbf{p}\sim\mathcal{N}(0,\mathbf{M})$, while the marginal distribution of the position variable is the target probability density, $\mathbf{x}\sim\pi$ [57].

The HMC sampling algorithm builds a Markov chain starting from an initial state $\mathbf{x}_0 = \mathbf{x}(0)$. Algorithm 1 summarizes the transition from the current Markov chain state $\mathbf{x}_k$ to a new state $\mathbf{x}_{k+1}$ [62]. Practical issues are related to the choice of the numerical integrator, the time step, and the choice of the function $\mathcal{J}(\mathbf{x})$ that represents the PDF we wish to sample from. The construction of the mass matrix $\mathbf{M}$ does not impact the final distribution, but does affect the computational performance of the algorithm [65]. The mass matrix $\mathbf{M}$ is symmetric and positive definite and is a parameter that is tuned by the user. It can be for

example, a constant multiple of the identity [60], or a diagonal matrix whose entries are the background error variances [64, 66]. We found that the latter approach is more efficient for the current application and used it in all numerical experiments reported here. We note that the generalized HMC method employed in [50] uses the dynamics of a modified system that is no longer Hamiltonian.

2.4 The Sampling Filter for Data Assimilation

The goal of this filter is to replace the analysis step in the traditional EnKF with a resampling procedure that draws representative ensemble members from the posterior distribution $\pi(\mathbf{x}) = \mathcal{P}^{\mathrm{a}}(\mathbf{x})$. Even if the posterior may in general be non-Gaussian we assume, as most of the current ensemble-based DA algorithms, that the posterior has the form:

$$\pi(\mathbf{x}) = \mathcal{P}^{\mathrm{a}}(\mathbf{x}) \propto \exp\left(-\mathcal{J}(\mathbf{x})\right), \tag{2.15a}$$

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2}\left\|\mathbf{x}-\mathbf{x}^{\mathrm{b}}\right\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\left\|\mathbf{y}-\mathcal{H}(\mathbf{x})\right\|^2_{\mathbf{R}^{-1}}, \tag{2.15b}$$

where $\mathbf{x}^{\mathrm{b}}$ is the background state (forecast), $\mathbf{y}$ is the observation vector, and $\mathcal{H}$ is the observation operator that is generally non-linear. Here the prior distribution is assumed Gaussian. When applying the sampling filter sequentially, this assumption is likely to be violated because the prior is a result of propagating the last (non-Gaussian) posterior forward to the forecast time. In this work, for simplicity, we assume that the prior can be well-approximated by a Gaussian distribution whose moments are inferred from the forecast ensemble. The use of more complex distributions like Gaussian mixtures to describe the prior is the topic of on-going work.

For sampling at time t_k the corresponding $\mathcal{J}(\mathbf{x})$ is:

$$\begin{aligned}\mathcal{J}(\mathbf{x}) &= -\log\left(\mathcal{P}^{\mathrm{a}}(\mathbf{x})\right) \\ &= \frac{1}{2}\left\|\mathbf{x}-\mathbf{x}^{\mathrm{b}}_k\right\|^2_{\mathbf{B}_k^{-1}} + \frac{1}{2}\left\|\mathbf{y}_k-\mathcal{H}_k(\mathbf{x})\right\|^2_{\mathbf{R}_k^{-1}},\end{aligned} \tag{2.16}$$

and its gradient has the form

$$\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}) = \mathbf{B}_k^{-1}\left(\mathbf{x}-\mathbf{x}^{\mathrm{b}}_k\right) - \mathbf{H}_k^T\,\mathbf{R}_k^{-1}\left(\mathbf{y}_k-\mathcal{H}_k(\mathbf{x})\right), \tag{2.17}$$

where $\mathbf{H}_k = \mathcal{H}'_k(\mathbf{x})$ is the linearized observation operator.

Algorithm 1 is used to generate $\mathrm{N_{ens}}$ ensemble members drawn from the posterior distribution $\{\mathbf{x}^{\mathrm{a}}_k(e) \sim \mathcal{P}^{\mathrm{a}}(\mathbf{x})\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$. The mean of this ensemble is an estimate of the *analysis*

Algorithm 1 HMC Sampling [62].

1: Draw a random vector $\mathbf{p}_k \sim \mathcal{N}(0, \mathbf{M})$.

2: Use a symplectic numerical integrator (from A.2) to advance the current state $(\mathbf{p}_k, \mathbf{x}_k)$ by a time increment T to obtain a *proposal* state $(\mathbf{p}^*, \mathbf{x}^*)$:

$$(\mathbf{p}^*, \mathbf{x}^*) = \mathcal{S}_T\left((\mathbf{p}_k, \mathbf{x}_k)\right), \tag{2.11}$$

where $\mathcal{S}$ refers to the symplectic integrator at hand.

3: Evaluate the loss of energy based on the Hamiltonian function. For the standard Verlet (A.2), two-stage (A.3), three-stage (A.4), and four-stage (A.5) integrators [63, 57] the loss of energy is computed as:

$$\Delta H = H(\mathbf{p}^*, \mathbf{x}^*) - H(\mathbf{p}_k, \mathbf{x}_k)\,. \tag{2.12}$$

For the Hilbert space integrator (A.6) [64] the loss of energy is computed as:

$$\begin{aligned} \Delta H &= \phi(\mathbf{x}^*) - \phi(\mathbf{x}_k) \\ &+ \frac{h^2}{8}\left(|\mathbf{M}^{-\frac{1}{2}}(-\nabla\phi(\mathbf{x}_k))|^2 - |\mathbf{M}^{-\frac{1}{2}}(-\nabla\phi(\mathbf{x}^*))|^2\right) \\ &+ h\sum_{i=1}^{m-1}\left(\mathbf{p}_k^T\left(-\nabla\phi(\mathbf{x}_k)\right)\right) \\ &+ \frac{h}{2}\left(\mathbf{p}_k^T\left(-\nabla\phi(\mathbf{x}_k)\right) + (\mathbf{p}^*)^T\left(-\nabla\phi(\mathbf{x}^*)\right)\right), \end{aligned} \tag{2.13}$$

where $\phi(\mathbf{x}) = -\log\left(\pi(\mathbf{x})\right)$, and $h = T/m$ is the integration time step.

4: Calculate the probability:

$$a^{(k)} = 1 \wedge e^{-\Delta H}. \tag{2.14}$$

5: Discard both $\mathbf{p}^*,\ \mathbf{p}_k$.

6: **(Acceptance/Rejection)** Draw a uniform random variable $u^{(k)} \sim \mathcal{U}(0, 1)$:

i- If $a^{(k)} > u^{(k)}$ accept the proposal as the next sample: $\mathbf{x}_{k+1} := \mathbf{x}^*$;

ii- If $a^{(k)} \leq u^{(k)}$ reject the proposal and continue with the current state: $\mathbf{x}_{k+1} := \mathbf{x}_k$.

7: Repeat steps 1 to 6 until sufficiently many distinct samples are drawn.

state, and the ensemble covariance estimates the analysis error covariance matrix. Note that the proposed sampling filter is not restricted to a specific form of the posterior PDF, and the Gaussian assumption (2.16) can in principle be removed. The remaining issue is to represent non-Gaussian probability density functions and their logarithm. In the next section we describe the proposed sampling filter as an alternative to the EnKF.

The sampling filter is described in Algorithm 2. Like most of the ensemble-based sequential DA algorithms the sampling filter consists of two stages, namely, the *forecast* step and the *analysis* step.

Start with an ensemble $\{\mathbf{x}^{\mathrm{a}}_{k-1}(e)\}_{e=1,\dots,\mathrm{N_{ens}}}$ describing the analysis PDF at time t_{k-1}. In the forecast step each ensemble member is propagated by the full model to the next time t_{k-1} where observations are available, resulting in the forecast ensemble. In the analysis step the HMC algorithm is simply used to sample from the posterior PDF of the state, providing the new analysis ensemble $\{\mathbf{x}^{\mathrm{a}}_{k}(e)\}_{e=1,\dots,\mathrm{N_{ens}}}$.

As stated in step *ii*) of Algorithm 2, the explicit representation of the matrix $\mathbf{B}_k$ is not necessary - one only needs to apply its inverse to a vector in (2.16), (2.17). Typically $\mathbf{B}_k$ is formed as a linear combination between a fixed matrix $\mathbf{B}_0$ and the ensemble covariance. The calculation requires to evaluate the products

$$\begin{aligned} u &= \mathbf{B}_k^{-1}\left(\mathbf{x}-\mathbf{x}^{\mathrm{b}}_k\right) \\ &= \left(\gamma\,\mathbf{B}_0 + \left(\frac{1-\gamma}{\mathrm{N_{ens}}-1}\sum_{e=1}^{\mathrm{N_{ens}}}\Delta\mathbf{x}(e)\left(\Delta\mathbf{x}(e)\right)^T\right)\circ\rho\right)^{-1}\left(\mathbf{x}-\mathbf{x}^{\mathrm{b}}_k\right), \end{aligned} \tag{2.18}$$

where $\Delta\mathbf{x}(e)$ is the deviation of the ensemble member $\mathbf{x}(e)$ from the mean of the ensemble, and ρ is the decorrelation matrix . The linear system

$$\left(\gamma\,\mathbf{B}_0 + \left(\frac{1-\gamma}{\mathrm{N_{ens}}-1}\sum_{e=1}^{\mathrm{N_{ens}}}\Delta\mathbf{x}(e)\left(\Delta\mathbf{x}(e)\right)^T\right)\circ\rho\right)\cdot u = \mathbf{x}-\mathbf{x}^{\mathrm{b}}_k, \tag{2.19}$$

can be solved without having to build the full matrix $\mathbf{B}_k$ as discussed in [67]. The linear system (2.19) is solved only once for each assimilation cycle to obtain the background term.

In our numerical experiments we build flow-dependent background error covariance matrices $\mathbf{B}_k$ at each time step. For experiments with Lorenz-96 model, a fully flow-dependent background error covariance matrix is used, i.e. $\gamma = 0$. In the experiments with the shallow water model on the sphere we chose $\gamma = 0.5$. We set $\mathbf{M}$ to be equal to the diagonal of $\mathbf{B}_k$ in case of Lorenz-96 model following [64, 66]. Taking $\mathbf{M}$ equal to the diagonal of

Algorithm 2 Sampling Filter

1: **Forecast step:** given an analysis ensemble $\{\mathbf{x}^{\mathrm{a}}_{k-1}(e)\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$ at time t_{k-1}; generate the forecast ensemble by via the model $\mathcal{M}$:

$$\mathbf{x}^{\mathrm{b}}_{k}(e) = \mathcal{M}_{t_{k-1}\rightarrow t_k}\left(\mathbf{x}^{\mathrm{a}}_{k-1}(e)\right), \quad e = 1, 2, \ldots, \mathrm{N_{ens}}.$$

2: **Analysis step:** given the observation vector $\mathbf{y}_k$ at time point t_k, follow the steps:

i- Set the initial state $\mathbf{x}_0$ of the Markov Chain to be to the best estimate available, e.g., the mean of the forecast ensemble. One can use the EnKF analysis if the cost is acceptable, and this choice is expected to result in a faster convergence of the chain to the stationary distribution.

ii- Calculate the ensemble-based forecast error covariance matrix $\mathbf{B}_k$ (and possibly balance it by a fixed (or frequently updated) covariance matrix $\mathbf{B}_0$), and apply localization as in equation (2.3d). It is important to emphasize that explicitly building the full background error covariance matrix is not necessary for the current algorithm to work.

iii- Choose a positive definite diagonal mass matrix $\mathbf{M}$. One choice that favors the performance of the sampling algorithm is the diagonal of the matrix $\mathbf{B}_k^{-1}$ [60] which scales the components of the state vector vary. Ideally, $\mathbf{M}$ should be set to the diagonal of the inverse posterior covariance matrix.

iv- Apply Algorithm 1 with initial state $\mathbf{x}_0$ and generate $\mathrm{N_{ens}}$ ensemble members. In practice one starts accepting samples after a warm-up phase (of, say, 30 steps), to guarantee that selected members explore the entire state space.

v- Use the generated samples $\{\mathbf{x}^{\mathrm{a}}_{k}(e)\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$ as an analysis ensemble and calculate the best estimate of the state (e.g. the mean), and the analysis error covariance matrix.

3: Increase time $k := k + 1$ and repeat steps 1 and 2.

$\mathbf{B}_k^{-1}$ leads to similar results for the Lorenz-96 model. For the shallow-water model on the sphere we set $\mathbf{M}$ to be equal to the diagonal of $\mathbf{B}_k^{-1}$.

2.4.1 Computational considerations

The sampling filter calculations shown in Algorithm 2 are computationally expensive. It is important to keep in mind that the main goal here is not to replace the operational filters when they work (the majority of situations). Instead we seek to develop a fully non-Gaussian sequential MCMC-based filter that is robust and can solve DA problems when the operational filters fail, at the expense of an *affordable increase* in the computational cost.

Moreover, as pointed out before, the sampling methodology can be used as replenishment tool in conjunction with parallel implementations of any practical filter, where some ensemble members are lost due to hardware failures. The forward model is not required during the sampling process (or the analysis step).

The cost of the analysis step of the sampling filter is determined by the settings of the chain parameters: the number of burn-in steps, the internal step size and the number of internal steps of the Hamiltonian trajectory, the number of states dropped at stationarity, and the ensemble size. The number of burn-in steps, the number of dropped states at stationarity, along with the size of the ensemble, define the length of the Markov chain. The number of internal steps is different for each symplectic integrator, and consequently the computational cost of the filter also dependents on the choice of the symplectic discretization method.

The total computational cost of the analysis step can be expressed in terms of linear algebra operations. Since the mass matrix is generally assumed to be diagonal, the computational bottleneck of the symplectic integrator is the evaluation(s) of the gradient of the potential energy. From Equation (2.17), assuming the observation operator is linear, three matrix-vector products (of dimension equal to the number of observations) and one linear system solution (of dimension equal to the number of states) are required to evaluate the gradient.

The total number of gradient evaluations for each symplectic integrator depends on the Hamiltonian trajectory settings. Verlet (A.2) requires one evaluation of the gradient of the cost functional (2.17) per step. If the length of the Hamiltonian trajectory is $T = m\,h$, where m is the number of steps of size h, then $3m$ matrix-vector products and m linear systems are evaluated to generate a proposal. An additional evaluation of the cost functional (2.16) is carried out per proposal to evaluate the loss of energy (2.12). An evaluation of

the cost functional (2.16) requires two additional dot-product evaluations. The evaluation of the kinetic energy term in the Hamiltonian requires matrix-vector multiplication by the diagonal mass matrix, which is the same as the cost of a vector dot-product. The cost of other integrators scales with the number of stages, i.e., is larger by a factor of two for (A.3), by a factor of three for (A.4), and by a factor of four for (A.3).

If the number of burn-in steps is b, and the number of states dropped at stationarity is $r-1$ (each r^{th} proposal is retained), then the total number of proposal states is $(b + r\mathrm{N}_{\mathrm{ens}})$. The total cost of the analysis step is $(b + r\mathrm{N}_{\mathrm{ens}})$ times the cost per step.

In the forecast step the forward model is run once for each ensemble member. The total number of forward model runs per assimilation cycle is then equal the size of the ensemble $\mathrm{N}_{\mathrm{ens}}$. The linear system (2.19) is solved once in order to update the background term.

In addition to the above discussion of the computational complexity of the sampling filter, we report the CPU times of both the sampling filter and the traditional filters in Section 2.5.11.

2.5 Numerical Results

2.5.1 The Lorenz-96 model

Numerical tests are primarily performed using the 40-variables Lorenz-96 model [68] which is described by the equations:

$$\frac{dx_i}{dt} = x_{i-1}\left(x_{i+1} - x_{i-2}\right) - x_i + F\,, \tag{2.20}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_{40})^T \in R^{40}$ is the state vector. The indices work in a circular fashion, e.g., $x_0 \equiv x_{40}$. The forcing parameter is set to $F = 8$ in our experiment. These settings make the system chaotic [69]. MATLAB implementations are used to carry out all the experiments reported in this chapter. A fourth order explicit Runge-Kutta scheme is used to propagate the Lorenz-96 model forward in time. One time step is $dt = 0.01$ (units). A vector of equidistant components ranging from -2 to 2 was integrated forward in time for 1000 time steps and the final state was taken as a reference initial condition for the experiments. The reference solution (truth) was created by integrating the reference initial condition forward in time over the timespan of each experiment. The initial background error covariance matrix $\mathbf{B}_0$ was created based on an error level of 8% of the magnitude of the reference initial condition. The decorrelation distance is specified to be 4. The

initial background state is obtained by adding normal random noise $r_0 \sim \mathcal{N}(0, B_0)$ to the reference initial condition. The initial ensemble was created by adding normal random noise $\sim \mathcal{N}(0, \mathbf{B}_0)$ to the initial background state. Synthetic observations are generated every 10 time steps by applying the observation operator to the reference trajectory and adding Gaussian noise such that the uncertainty level in observation is 5%. See Appendix A.1 for details on how the initial background error and the observation error covariance matrices are created. To study the behavior of the sampling filter with small ensemble, the number of ensemble members is chosen to be 30 for all experiments with Lorenz-96 model.

2.5.2 Observations and observation operators

We tested the performance of the sampling filter with several observation operators of different complexities and varying levels of non-linearity. In this section, we show the results of a linear, a discontinuous-quadratic, and an exponential observation operators. The discontinuous quadratic observation (quadratic with a threshold) operator used here was employed by Zupanski [23, 16] in the simple case of one dimensional state space. For experiments with more observation operators with different levels of discontinuity and nonlinearity see [70].

a) **Linear observation operator:** in all the experiments conducted with Lorenz-96 model we assume that third components only are observed. The linear observation operator takes the form:

$$\mathcal{H}(\mathbf{x}) = \mathbf{H}\mathbf{x} = (x_1,\ x_4,\ x_7,\ \ldots,\ x_{37},\ x_{40})^T \in \mathbb{R}^{14}\,, \tag{2.21}$$

This operator is provided to compare compare the performance of the sampling filter with EnKF which is known to be optimal in this case.

b) **Quadratic observation operator with a threshold:** this observation operator is similar to the simple version used by Zupanski et al in [16]. The observation vector is:

$$\mathcal{H}(\mathbf{x}) = (x'_1,\ x'_4,\ x'_7,\ \ldots,\ x'_{37},\ x'_{40})^T \in \mathbb{R}^{14}\,, \tag{2.22}$$

where

$$x'_i = \begin{cases} x_i^2 & : x_i \geq 0.5 \\ -x_i^2 & : x_i < 0.5\,. \end{cases} \tag{2.23}$$

This operator is non-linear and discontinuous.

c) **Exponential observation operator:** this is a highly nonlinear, differentiable observation operator:

$$\mathcal{H}(\mathbf{x}) = (e^{r\cdot x_1},\ e^{r\cdot x_4},\ e^{r\cdot x_7},\ \ldots,\ e^{r\cdot x_{37}},\ e^{r\cdot x_{40}})^T \in \mathbb{R}^{14}, \tag{2.24}$$

where $r \in \mathbb{R}$ is a scaling factor that controls the degree of nonlinearity.

2.5.3 Experimental setting

The sampling filter can be tuned to accommodate complex settings of the experiment in hand. The parameters of the Hamiltonian trajectory can be tuned empirically such as to achieve a specific acceptance rate. In general the step size should be chosen to ensure that the rejection rate falls between 25% and 30%, an approach we followed in our settings, and the number of steps should be large [60]. We chose the length of the Hamiltonian trajectory of the sampling filter empirically to be $T = 10$ with $h = 0.01$ and $m = 10$ for all symplectic integrators. This choice is made by trial and error such that the sampling filter with the standard position Verlet integrator yields seemingly satisfying results with the linear observation operator at moderate cost. More precisely, these settings lead to acceptance rates from 25% to 30% when a linear observation operator is used in the present experiments. In general, however, the time-stepping parameters of each symplectic integrator should be set individually to get the best performance of the sampling filter. The goal here is to show that even with a very simple setting of the sampling filter parameters one can obtain promising results for this experimental setting. The mass matrix $\mathbf{M}$ is set as a diagonal matrix which diagonal is the precisions of the background errors at each time step. To guarantee that the Markov chain reaches the stationary distribution before starting the sampling process a set of 50 steps are perform as burn-in stage. We noticed however that the chain always converges in a small number of burn-in steps, typically 10-20 steps. Stationarity tests will be given special attention in our future work. An alternative to the burn-in steps is to apply a three dimensional variational (3D-Var) step which will certainly result in a state belonging to the posterior distribution. That resulting state can be used to initialize the chain, and sampling can start immediately. After the burn-in stage an ensemble member is selected after each 10 generated states. dropping states between successive selection has the effect of decreasing correlation between generated ensemble members since the chain is not memoryless. The number of generated states (between successive selections) that are not retained during the sampling from stationarity will be referred to as the number of mixing steps. In our experiments the acceptance probability is high (usually over 0.9) with this sampling strategy. The number of mixing steps is a parameter that can be tuned by the user to control the performance of the sampling filter.

Stability requirements impose tight upper bounds on the step size h of the Verlet integrator. The step size should be decreased with the increasing dimension of the system in order to maintain $O(1)$ acceptance probability [71]. On the other hand long Hamiltonian trajectories (large steps of the symplectic integrator) are needed in order to explore the space efficiently. There is no precise rule available to select the optimal step size values [62] and consequently h, and m should be tuned for each problem. The higher-order integrators (A.3), (A.4), (A.5) are expected to be more stable than Verlet for larger step sizes [63, 60] and a smaller number of steps.

To guarantee ergodicity of the Markov chain, which is a property required for the chain to converge to its invariant distribution, we follow [63, 60] and change the step length at the beginning of each Markov step (once at the beginning of the Hamiltonian trajectory) to $h = (1 + r)\, h_{\text{ref}}$ where h_{ref} is a reference step size and $r \sim \mathcal{U}(-0.2, 0.2)$ is a uniformly distributed random variable. Randomizing the step size of the symplectic integrator, in addition to other benefits, ensures that the results obtained are not entrusted with specific choice of the step size [60].

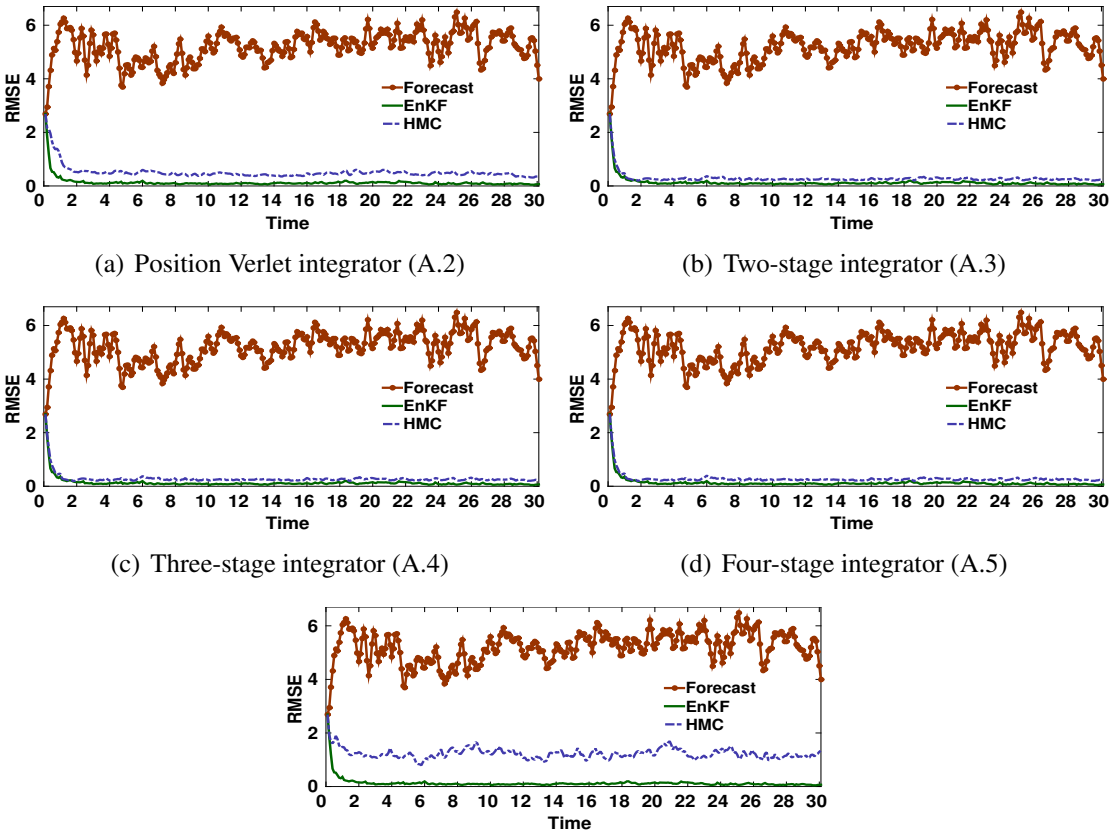
Each numerical experiment performs 100 realizations of the sampling filter. Each realization uses the same settings but the sequence of random numbers generated by the sampling filter, for both the potential variable and the acceptance/rejection rule, was different. The root mean squared error (RMSE) metric is used to compare the analyses against the reference solution at observation time points:

$$\textbf{RMSE} = \sqrt{\frac{1}{\mathrm{N}_{\text{state}}} \sum_{i=1}^{\mathrm{N}_{\text{state}}} (x_i - x_i^{\text{true}})^2}\,, \tag{2.25}$$

where $\mathbf{x}^{\text{true}}$ is the reference state of the system and $\mathrm{N}_{\text{state}} = 40$ in case of Lorenz-96 model. The RMSE is calculated at all assimilation time points along the trajectory over the time span of the experiment. The average of RMS errors over the 100 realizations of the sampling filter is plotted against RMS errors obtained from the traditional filters in Figures 2.1, 2.3, 2.5, 2.9, and 2.11. The RMS error statistics are presented in Tables 2.1, 2.2, and 2.3. In addition to RMSE, we use the rank histograms (Talagrand diagrams) [72, 73] to assess the quality of the generated ensembles.

2.5.4 Linear observation operator experiments

Figure 2.1 shows the average RMSE of the 100 instances of the sampling filter and the RMSE obtained by applying EnKF (with inflation factor 1.09). Statistics of RMSE results



(a) Position Verlet integrator (A.2)

(b) Two-stage integrator (A.3)

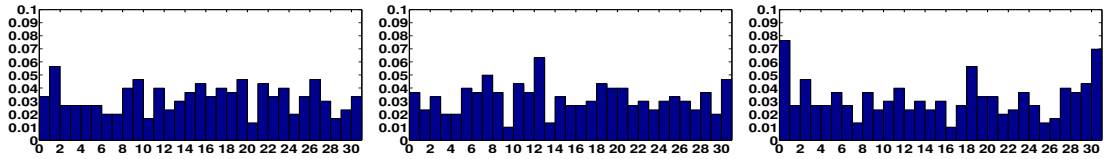
(c) Three-stage integrator (A.4)

(d) Four-stage integrator (A.5)

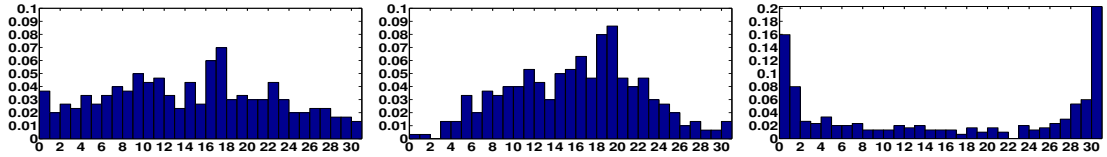
(e) Integrator defined on Hilbert space (A.6)

Figure 2.1: Data assimilation results with the Linear operator (2.21). The symplectic integrator used is indicated under each panel. The time step for all symplectic integrators is $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps. The RMSE reported for the sampling filter (HMC) is the average taken over the 100 realizations of the filter.

obtained by applying MLEF and IEnKF are summarized in Table 2.1. Rank histograms of the first two components of the state vector along the 300 assimilation cycles are given in Figure 2.2. We present rank histograms only for the first two components of the state vector because we noticed that rank histograms of observed components look very similar, and unobserved components nearly have similar rank histograms. Using the standard po-



(a) x_1; Position Verlet integrator (A.2) (b) x_1; Two-stage integrator (A.3) (c) x_1; Integrator defined on Hilbert space (A.6)



(d) x_2; Position Verlet integrator (A.2) (e) x_2; Two-stage integrator (A.3) (f) x_2; Integrator defined on Hilbert space (A.6)

Figure 2.2: Data assimilation results with the Linear operator (2.21). The symplectic integrator used is indicated under each panel. The time step for all symplectic integrators is $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps. The rank histograms are shown for the first two components of the state vector over 300 assimilation cycles compared to the truth. The plotted component is indicated under each panel.

sition Verlet as the symplectic integrator results in occasional failures (Figure 2.1). These failures are mostly due to the untuned step settings. On the other hand the higher order integrators show quite good performance. From Figure 2.2 we also see that using position Verlet the observed components (e.g. x_1) and unobserved components (e.g. x_2) tend to have uniform rank histogram while the use of higher order integrators results in overdispersed ensemble on average. Although not shown here, rank histograms resulting from three-stage and four-stage integrators are almost identical to those obtained by using two-stage integrator. The truth however lies in the middle of the ensemble and one can conclude that shorter Hamiltonian trajectory can be used in the case of higher-order integrators while the standard Verlet integrator requires smaller step size and larger number of steps in order to maintain stability and to reach distant points in state space respectively. For example, if the length of the Hamiltonian trajectory $T = 0.1$ is kept fixed, and the step size h is reduced to 0.005, the likelihood of filter diverging (see the maximum RMSE in

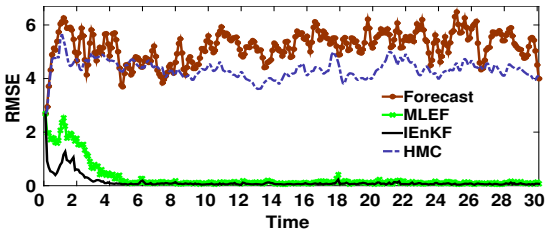
Table 2.1 for this case) can be drastically reduced. This is confirmed by the results reported in Tables 2.1 and 2.2, where the standard deviation of RMSE is reduced from 0.546054 to 0.039302, and the RMSE average is slightly decreased as well. Further tuning of the filter parameters can lead to smaller RMSE, and consequently result in a better estimation of the true state. However, in nonlinear settings, one is concerned more about the distribution of the ensemble members under the posterior PDF, rather than estimating one specific state.

The use of the infinite dimensional integrator (A.6) (defined on Hilbert space) results in under-dispersive ensemble however it does not show signs of divergence. This is illustrated in Figures 2.1(e), 2.2(c), and 2.2(f).

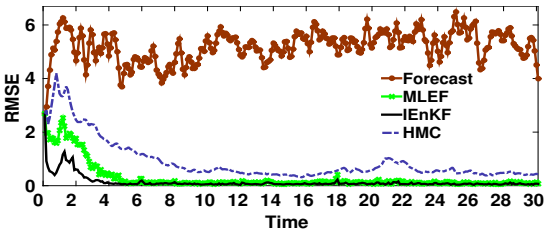
2.5.5 Quadratic observation operator with threshold experiments

Figure 2.3 shows the RMSE results with quadratic observation operator (2.22) with threshold $a = 0.5$. In this case EnKF failed to converge, while both MLEF and IEnKF show very good behavior. However, we noticed that to get MLEF to produce good results inflation was needed. The performance of MLEF in the current setting is sensitive to both the observation frequency (the dimension of the observation space) and the noise level.

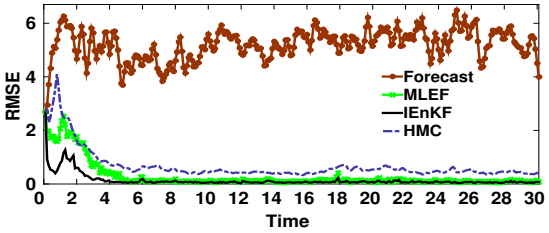
For clarity the figures only report results of the sampling filter (HMC) along with results obtained using MLEF (with inflation factor 1.25) and IEnKF (with inflation factor 1.09). Statistics of RMSE results for EnKF, MLEF, IEnKF and the sampling filter are given in Table 2.1. Given the current untuned parameter settings, we believe that the sampling filter using Verlet integrator fails due to the high non-linearity of the observation operator and/or the uncertainty levels. The high order integrators show good results and the analysis RMSE is close to the level obtained in case of linear observation operator. The likelihood of outliers is small and decreases using higher-order integrators as can be seen from the maximum values of the RMSE shown in Table 2.1. The Hilbert integrator gives reasonable results. Further tuning of the filter parameters can result in much better results.
Rank histograms (2.4), and standard deviations of RMS errors in Table 2.1, both show that higher order integrators, e.g. three-stage (Figures 2.4(a), 2.4(b)) produce acceptable spread around the truth. The integrator defined on Hilbert space continues to show under-dispersive behaviour with larger variation of the RMSE. Careful tuning of the step size used in case of the position Verlet and the integrator defined on Hilbert space is required for the sampling filter to converge. For example, setting the length of the Hamiltonian trajectory to $T = 0.03$ with $h = 0.001$, $m = 30$ results in better performance as shown in Figures 2.5, 2.6, and in Table 2.2. The unobserved components have roughly uniform shapes except for the two spikes at the two extremes of the rank histogram in Figure 2.6(b).



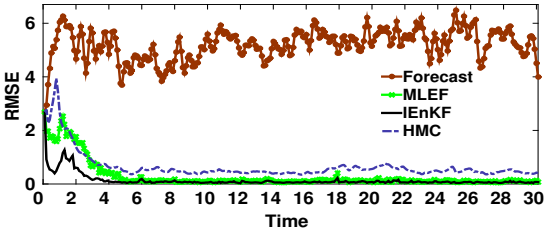
(a) Position Verlet integrator (A.2)



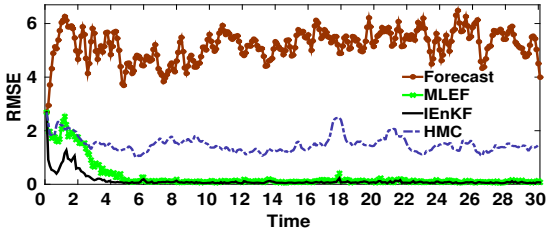
(b) Two-stage integrator (A.3)



(c) Three-stage integrator (A.4)

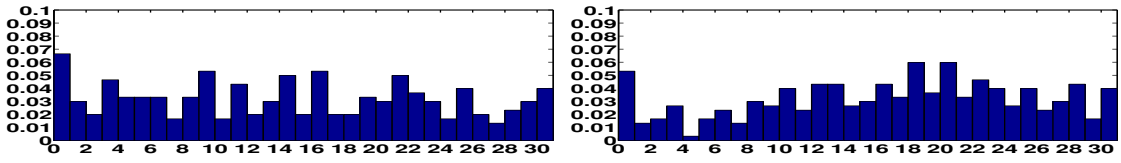


(d) Four-stage integrator (A.5)

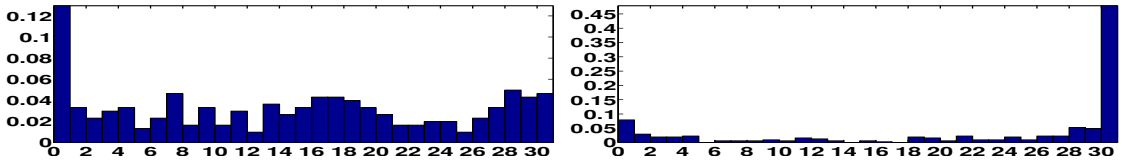


(e) Integrator defined on Hilbert space (A.6)

Figure 2.3: Data assimilation results with the quadratic observation operator (2.22) with a threshold $a = 0.5$. The symplectic integrator used is indicated under each panel. The time step for all symplectic integrators is $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps. The RMSE reported for the sampling filter (HMC) is the average taken over the 100 realizations of the filter.



(a) x_1; Three-stage integrator (A.4) (b) x_2; Three-stage integrator (A.4)



(c) x_1; Integrator defined on Hilbert space (A.6) (d) x_2; Integrator defined on Hilbert space (A.6)

Figure 2.4: Data assimilation results with the quadratic observation operator (2.22) with a threshold $a = 0.5$. The symplectic integrator used is indicated under each panel. The time step for all symplectic integrators is $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps. The rank histograms are shown for the first two components of the state vector over 300 assimilation cycles compared to the truth. The plotted component is indicated under each panel.

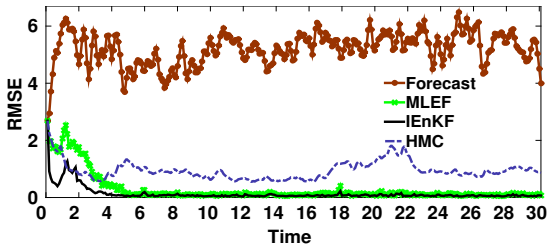
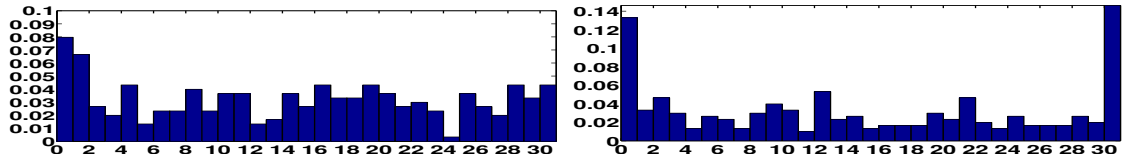


Figure 2.5: Data assimilation results with the quadratic observation operator (2.22) with a threshold $a = 0.5$. The symplectic integrator used is position Verlet with time step $T = 0.03$ with $h = 0.001$, $m = 30$, and 10 mixing steps. The RMSE reported for the sampling filter (HMC) is the average taken over the 100 realizations of the filter.



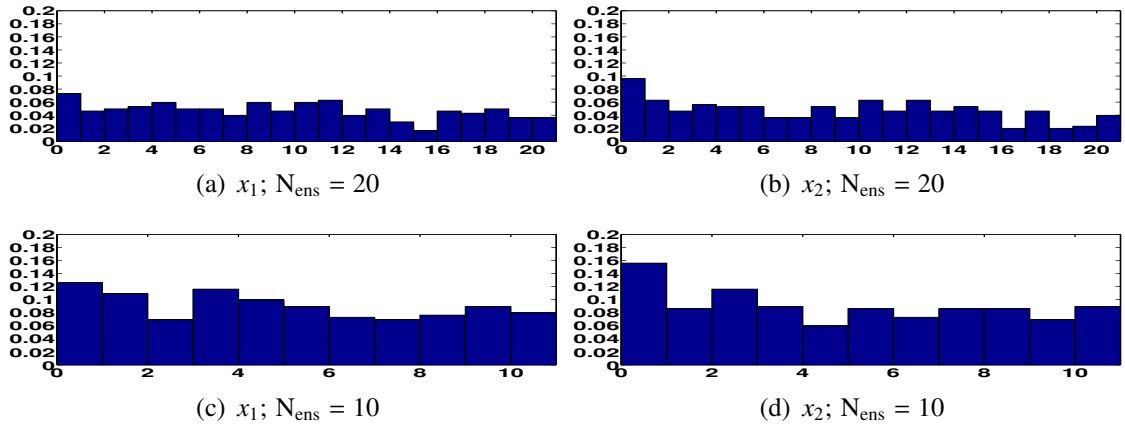
(a) x_1 (b) x_2

Figure 2.6: Data assimilation results with the quadratic observation operator (2.22) with a threshold $a = 0.5$. The symplectic integrator used is position Verlet with time step $T = 0.03$ with $h = 0.001$, $m = 30$, and 10 mixing steps. The rank histograms are shown for the first two components of the state vector over 300 assimilation cycles compared to the truth. The plotted component is indicated under each panel.

These results indicate that a longer Hamiltonian trajectory might be needed in order for the sampling filter to be able to sample from the tails of the posterior distribution.

The ensemble size is taken to be $\mathrm{N_{ens}} = 30$, however it is important to discuss the filter performance in the case where fewer ensemble members are desirable. Reducing the ensemble size definitely reduces the cost of the sampling filter but increases the sampling error introduced. The spread of the ensemble should not be destroyed where only few ensembles are used, otherwise the filter will be questionable. The sampling filter was tested in several settings where the ensemble size is varied. We pick the case where three-stage symplectic integrator (with $h = 0.01$, $m = 10$) is used in the presence of the discontinuous quadratic observation operator (2.22) to show results of the sampling filter with different ensemble sizes. As appreciated from Figure 2.7 the spread of the ensemble with 20, 10 members is preserved compared to results with 30 ensemble members shown in Figure 2.4(a), 2.4(b) giving a hope that in case of large dimensional systems, small number of ensemble members can produce sufficient information of the posterior distribution. When the ensemble size is varied, all other components of the state vector show similar behavior to the case where 30 ensemble members are used but we showed only the first two components to be consistent with results shown before.

The sampling filter can be used to replenish the lost ensemble members in parallel implementations of the traditional filters (e.g EnKF, IEnKF). The traditional approach to replenishing is to perturb the average of the remaining ensemble members with random noise generated from a Gaussian distribution with zero mean and covariance matrix approximated based on the remaining ensemble members. This approach leads to the new forecast states living in the subspace spanned by the remaining ensemble members, a fact that can lead to filter degradation in real applications. Moreover, this strategy is no longer



(a) x_1; $N_{ens} = 20$ (b) x_2; $N_{ens} = 20$

(c) x_1; $N_{ens} = 10$ (d) x_2; $N_{ens} = 10$

Figure 2.7: Data assimilation results with the quadratic observation operator (2.22) with a threshold $a = 0.5$. The symplectic integrator used is the three-stage (A.4) with time step $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps. Rank histograms of the first two components of the state vector are shown where ensemble size is varied. Plotted components and ensemble size are shown under each panel.

be valid if the Gaussianity assumption of the prior is violated.

One way to address this problem is to generate ensembles from the analysis in the previous cycle using HMC sampling filter, and to propagate them forward to the current time instance. In this case the sampling filter chain can be started from the average of the analysis ensemble and no burn-in states are required. We believe that the use of the sampling filter for replenishing dead ensemble members can be advantageous in the cases where few ensemble members are used or if the probability of node failure is high. Generally speaking, replenishment using the sampling filter does not depend the probability of node failure.

To illustrate this, we perform a simulation with the Lorenz-96 model where each of the 30 ensemble members is killed with a (admittedly large) probability $P_f = 0.25$ (Figure 2.8). For clarity we removed the RMSE results for the case when the ensemble members are not replenished at all whenever any member dies out.

When the ensemble is not replenished, the IEnKF diverges after approximately 10 to 15 cycles. Replenishing the ensemble by perturbing the average of the remaining ensemble members with Gaussian perturbations with zero mean and covariance matrix $\mathbf{B}_k$ still can lead to filter divergence, as shown in Figure 2.8. Even in this case the sampling filter successfully replenishes the ensemble and aids IEnKF to maintain its desired performance.

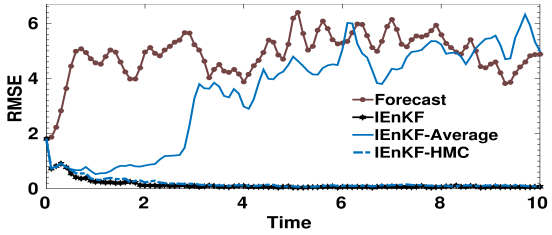


Figure 2.8: Data assimilation results with the quadratic observation operator (2.22) with a threshold $a = 0.5$. Probability of node failure is assumed to be 0.25 and the ensemble members are replenished by perturbing the ensemble average (IEnKF-Average), and using the sampling filter (IEnKF-HMC). IEnKF refers to RMSE results with no members' failures (i.e. $P_f = 0$). The two-stage symplectic integrator used with time step $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps.

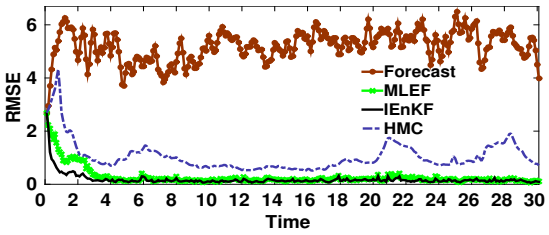
2.5.6 Exponential observation operator (with factor $r = 0.2$) experiments

Figure 2.9 shows the RMSE results with the exponential observation operator (2.24) with factor $r = 0.2$. This observation operator is differentiable, however small perturbations in the state might result in relatively large changes in the measured values. Under strongly nonlinear conditions the sampling filter performs better than EnKF and is giving results comparable to results from MLEF and IEnKF. MLEF results are obtained by forcing inflation with factor 1.15 in this case. Based on RMSE results, it is quite clear that the best choice is the three-stage symplectic integrator for the given parameter settings (Figures 2.9(c), 2.10). The other integrators require longer Hamiltonian trajectories or smaller step size or both to sample the posterior properly.

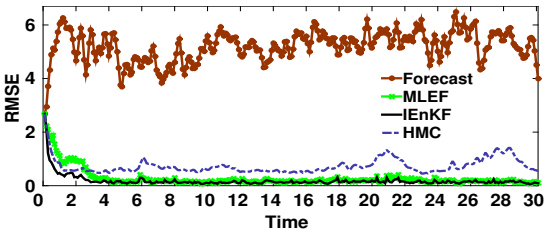
2.5.7 A highly nonlinear observation operator

We have also tested the sampling filter capabilities in a very challenging setting: the exponential observation operator (2.24) is considered with a factor of $r = 0.5$. This factor leads to a large range of observation values (from $e^{-3.7}$ to $e^{6.2}$). In addition, small perturbations in the state variables cause very large changes in the corresponding measurements. EnKF, MLEF, and IEnKF all diverged when applied to this test, and consequently their results are not reported here.

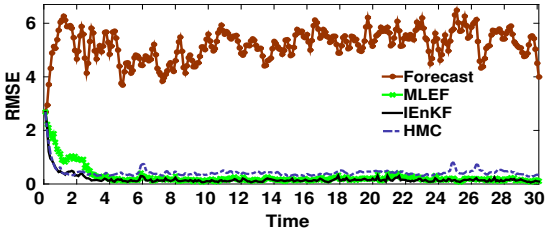
This test problem is challenging for the sampling filter as well and the symplectic integra-



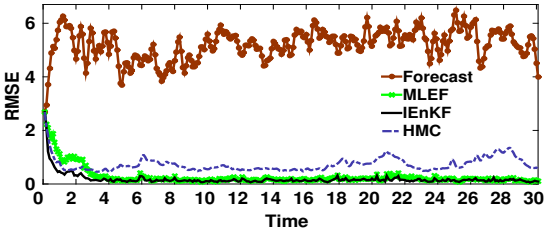
(a) Position Verlet integrator (A.2)



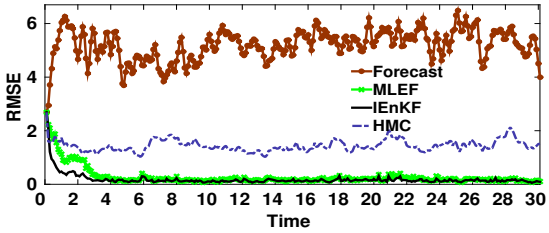
(b) Two-stage integrator (A.3)



(c) Three-stage integrator (A.4)

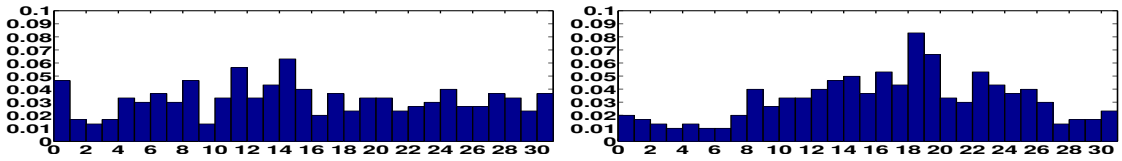


(d) Four-stage integrator (A.5)



(e) Integrator defined on Hilbert space (A.6)

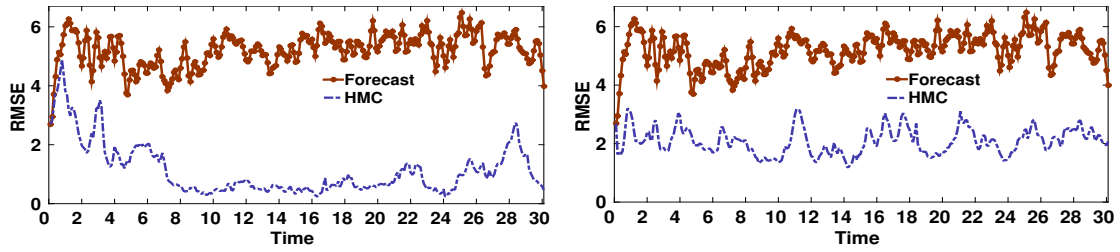
Figure 2.9: Data assimilation results with the exponential observation operator (2.24) with factor $r = 0.2$. The symplectic integrator used is indicated under each panel. The time step for all symplectic integrators is $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps. The RMSE reported for the sampling filter (HMC) is the average taken over the 100 realizations of the filter.



(a) x_1 (b) x_2

Figure 2.10: Data assimilation results with the exponential observation operator (2.24) with factor $r = 0.2$. The symplectic integrator used is indicated under each panel. The symplectic integrator used is the three-stage (A.4) with time step $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps. The rank histograms are shown for the first two components of the state vector over 300 assimilation cycles compared to the truth. The plotted component is indicated under each panel.

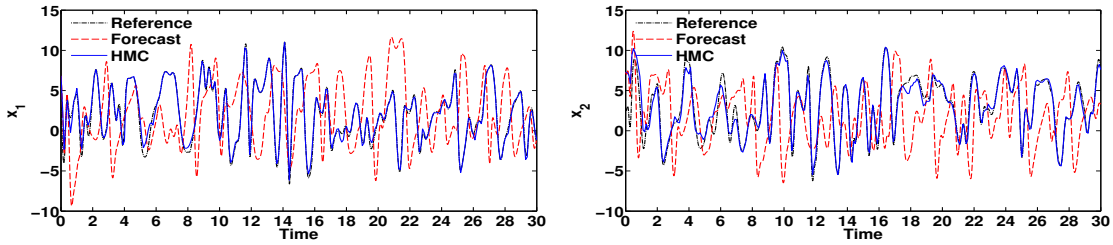
tion step sizes need to be tuned to achieve convergence. For example, the number of steps taken by three-stage integrator had to be increased to $m = 60$ while keeping the step-size fixed to $h = 0.01$, to result in good performance as shown in Figure 2.11(a). The length of the trajectory of the Hamiltonian system has to be increased as well. For the infinite dimensional integrator, a shorter trajectory of the Hamiltonian system works well if the step size is sufficiently reduced, e.g., $h = 0.001$, and $m = 30$ as shown in Figure 2.11(b). Empirical tuning of the Verlet, two-stage, and four-stage integrators proved to be challenging with this observation operator. The computational cost associated with the tuned three-



(a) Three-stage integrator (A.4); $h = 0.01$, $m = 60$ (b) Integrator defined on Hilbert space (A.6); $h = 0.001$, $m = 30$

Figure 2.11: Data assimilation results with the exponential observation operator (2.24) with a factor $r = 0.5$. The symplectic integrator used is indicated under each panel. The step size h and the number of steps m are indicated under each panel. The number of mixing steps is 30. The RMSE reported for the sampling filter (HMC) is the average taken over the 100 realizations of the filter.

stage integrator in the present experiment is much higher than the cost resulting from the use of the infinite dimensional integrator. Despite the relatively large RMSE revealed by Figure 2.11(b), Figure 2.12 shows that the analysis produced by the HMC sampling filter using the integrator defined on Hilbert space follows the truth reasonably closely, which can justify the argument that this level of RMSE can be accepted if the traditional filters fail and the cost associated with the higher order integrators is prohibitive. The Hilbert integrator operates at a much lower cost, and can be used to periodically check the results obtained with the three-stage integrator to safeguard against outliers.



(a) x_1 (b) x_2

Figure 2.12: Data assimilation results with the exponential observation operator (2.24) with a factor $r = 0.5$. The symplectic integrator used is the integrator defined on Hilbert space (A.6) with time step $T = 0.03$ with $h = 0.001$, $m = 30$, and 30 mixing steps. The first two components of the state vector are plotted.

The statistics of the results with Lorenz-96 model are summarized in Tables 2.1, 2.2, and 2.3. Given the unified time step settings $h = 0.01$, $m = 10$ for all symplectic integrators, Table 2.1 summarizes results for the linear observation operator (2.21), the discontinuous quadratic observation operator (2.22), and the exponential observation operator (2.24) with factor $r = 0.2$. Table 2.2 summarizes results for the linear observation operator (2.21), and the discontinuous quadratic observation operator (2.22) where the step settings of the position Verlet integrator were tuned to give better results. Table 2.3 summarizes results for the exponential observation operator (2.24) with factor $r = 0.5$. The results shown are computed for 100 instances of the sampling filter, EnKF, MLEF, and IEnKF, over the last 20% of the assimilation cycles for each experiment.

Table 2.1: RMS error statistics of experiments for assimilation time points $24 \leq t \leq 30$ (after filter stabilizes). The time step for all symplectic integrators is $T = 0.1$ with $h = 0.01$, $m = 10$, and 10 mixing steps.

Observation Operator	**Statistics**	**Symplectic integrator used with sampling filter**					**Traditional Filters**		
		Verlet	**Two-stage**	**Three-stage**	**Four-stage**	**Hilbert**	**EnKF**	**MLEF**	**IEnKF**
Linear observation operator	Min	0.282158	0.225048	0.221871	0.22929	1.126023	0.04631	0.027045	0.027835
	Max	4.553241	0.331516	0.275494	0.292533	1.257923	0.13634	0.124409	0.197427
	Mean	0.433644	0.250042	0.249086	0.252403	1.179563	0.079809	0.069438	0.080403
	Std	0.546054	0.014128	0.01077	0.012136	0.027446	0.020084	0.021752	0.027741
Quadratic observation operator with threshold	Min	3.51313	0.362567	0.370246	0.364221	1.203196	2.223534	0.048766	0.033307
	Max	5.285517	2.525502	0.607215	1.123681	1.889033	8.452283	0.175145	0.116735
	Mean	4.344827	0.476498	0.444522	0.462515	1.394191	3.949765	0.094103	0.06193
	Std	0.318406	0.218234	0.043667	0.086601	0.134929	1.184631	0.028429	0.017315
Exponential observation operator with $r = 0.2$	Min	0.527899	0.478779	0.389757	0.500551	1.332802	3.598643	0.065779	0.05316
	Max	2.406581	2.138528	0.596158	1.874854	1.840191	7.734853	0.239167	0.240487
	Mean	1.119951	0.902746	0.446232	0.887058	1.546304	5.381176	0.155157	0.132423
	Std	0.45629	0.374384	0.034703	0.35285	0.099781	0.956734	0.041048	0.036759

Table 2.2: RMS error statistics of experiments for assimilation time points $24 \leq t \leq 30$ (after filter stabilizes). Step parameters of the position Verlet symplectic integrator are tuned to give acceptable results.

Statistics	**Observation operator and symplectic integrator used**	
	Linear $\mathcal{H}$; Verlet: $h = 0.001$, $m = 30$	**Discontinuous quadratic $\mathcal{H}$;** Verlet: $h = 0.005$, $m = 20$
Min	0.291365	0.580504
Max	0.569277	1.710029
Mean	0.362358	0.900346
Std	0.039302	0.370047

Table 2.3: RMS error statistics of experiments for assimilation time points $8 \leq t \leq 10$ (after filter stabilizes). The exponential observation operator with factor $r = 0.5$ is used.

Statistics	**Symplectic integrator used with sampling filter**	
	Three-stage; $h = 0.01$, $m = 60$	**Hilbert;** $h = 0.001$, $m = 30$
Min	0.304178	1.234498
Max	2.671971	2.350684
Mean	0.439776	1.699096
Std	0.274643	0.250088

2.5.8 Tuning the sampling filter parameters

Tuning the sampling filter parameters can prevent outliers (filter divergence) that can happen especially when nonlinear observation operators are used. In addition to selecting the mass matrix $\mathbf{M}$ and the number of burn-in steps, there are two more parameters to be tuned. The first parameter is the length of the Hamiltonian trajectory (including step size and the number of steps of the symplectic integrator). The second parameter to be tuned is the number of steps skipped between selected states at stationarity (referred to as mixing steps). A common and a simple approach is to tune the parameters of the symplectic integrator by monitoring the acceptance rate in a preprocessing step. For simplicity, we followed this strategy in the present work. Automatically tuned versions of HMC have been proposed recently as well. No-U-Turn sampler (NUTS) [74] is a version of HMC capable of automatically tuning its parameters by prohibiting the sampler from retracing its steps along the constructed Hamiltonian trajectory. Riemann manifold HMC (RMHMC) [65] is another HMC sampler that tunes its parameters automatically using third-order derivative information. Tuning the symplectic integrator step settings can enhance the filter performance and reduce the of filter divergence, however, it might not be sufficient to completely prevent outliers. Tuning the number of mixing steps can in principle greatly enhance both the performance of the filter and the reliability of the results. The number of mixing steps in all experiments with Lorenz-96 is empirically set to 10 but it is not optimal. Increasing the number of mixing steps might be tempting, however, we cannot conclude that skipping more states will increase the chance of filter convergence [70]. A careful tuning of both the step size and the number of mixing steps in the chain may overcome the problem of outliers and lead to the desired performance of the filter. Generalized HMC with circular matrix can also be considered to shorten the decorrelation time [50] and hence reduce the number of mixing steps required by the sampling filter. From a statistical perspective, the former parameters can be tuned for example by inspecting the acceptance rate of the proposed states and the effective size of the ensemble.

In addition to controlling the time step settings of the integrator, and tuning the number of steps of the chain, we can use the Hilbert integrator (with tuned step size) to periodically validate the ensembles obtained using other integrators, since the Hilbert integrator suffers less from outliers. A simple solution is to run the assimilation process several times and exclude outlier states by creating a combined ensemble. Care must be exercised, however, to not change the probability density. These alternatives will be inspected in depth in future work in the context of more complex models.

2.5.9 Shallow water model on a sphere

As a first step towards large models we test the proposed sampling filter on the shallow water model on a sphere, using linear observation operator where all components are observed. The shallow water equations provide a simplified model of the atmosphere which describes the essential wave propagation mechanisms found in general circulation models (GCMs) [75]. The shallow water equations in spherical coordinates are given as

$$\frac{\partial u}{\partial t} + \frac{1}{a\cos\theta}\left(u\frac{\partial u}{\partial \lambda} + v\cos\theta\frac{\partial u}{\partial \theta}\right) - \left(f + \frac{u\tan\theta}{a}\right)v + \frac{g}{a\cos\theta}\frac{\partial h}{\partial \lambda} = 0, \tag{2.26a}$$

$$\frac{\partial v}{\partial t} + \frac{1}{a\cos\theta}\left(u\frac{\partial v}{\partial \lambda} + v\cos\theta\frac{\partial v}{\partial \theta}\right) + \left(f + \frac{u\tan\theta}{a}\right)u + \frac{g}{a}\frac{\partial h}{\partial \theta} = 0, \tag{2.26b}$$

$$\frac{\partial h}{\partial t} + \frac{1}{a\cos\theta}\left(\frac{\partial (hu)}{\partial \lambda} + \frac{\partial (hv\cos\theta)}{\partial \theta}\right) = 0\,. \tag{2.26c}$$

The Coriolis parameter is given by $f = 2\Omega\sin\theta$, where Ω is the angular speed of the rotation of the Earth, and θ is latitudinal direction. The longitudinal direction is λ. The height of the homogeneous atmosphere is represented by h, the zonal and meridional wind components are given by u and v respectively. The radius of the earth is a, and the gravitational constant is given by g. The space discretization follows the unstaggered Turkel-Zwas scheme [76]. The discretization has $nlon = 72$ nodes in longitudinal direction and $nlat = 36$ nodes in the latitudinal direction. The semi-discretization in space results in the following discrete model:

$$\mathbf{x}_{k+1} = \mathcal{M}_{t_k \to t_{k+1}}(\mathbf{x}_k, \theta), \quad k = 0, \ldots, N; \qquad \mathbf{x}_0 = \mathbf{x}_0(\theta)\,. \tag{2.27}$$

The state space vector $\mathbf{x}$ in (2.27) combines the zonal wind, the meridional wind, and the height variables into the vector $\mathbf{x} \in \mathbb{R}^{\mathrm{N}}_{\mathrm{state}}$ with $\mathrm{N}_{\mathrm{state}} = 3 \times \mathrm{nlat} \times \mathrm{nlon} = 7776$. The time integration is conducted using an adaptive time-stepping algorithm. A reference initial condition is used to generate a reference trajectory. Synthetic observations are created from the reference trajectory by adding Gaussian noise with zero mean and fixed standard deviation for each of the three components. The level of observation noise for height component is set to 1.5% of the average magnitude of the reference height component in the reference initial condition. The level of observation noise for wind components is set to 10% of the average magnitude of the reference wind component in the initial condition. Based on these uncertainty levels, the standard deviations of observation errors are $\sigma_h = 700$, $\sigma_u = \sigma_v = 3$ for the height, the zonal and the meridional wind components respectively. The initial background state is created by perturbing the reference initial condition by a Gaussian noise drawn from a modeled background error covariance matrix

$\mathbf{B}_0$. The standard deviation of the background errors for the height component is 2% of the average magnitude of the reference height component in the reference initial condition. The standard deviation of the background errors for the wind components is 15% of the average magnitude of the reference wind component in the reference initial condition.

The modeled version of the background error covariance, $\mathbf{B}_0$, that accounts for correlations between state variables is created as follows:

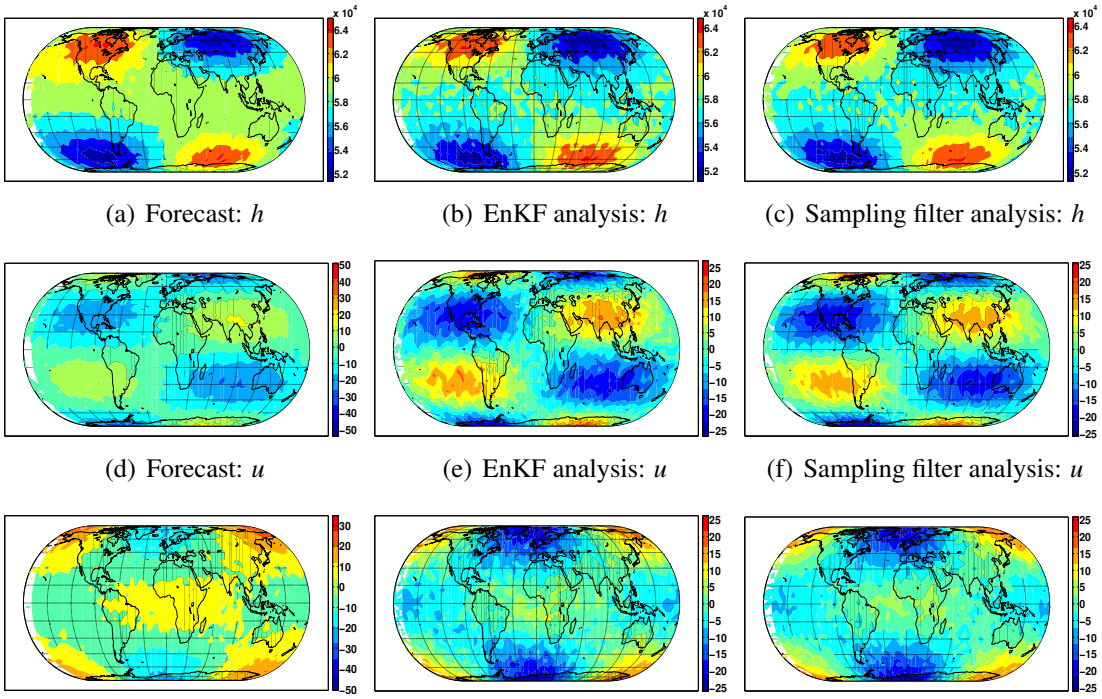
- Start with a diagonal background error covariance matrix with uncertainty levels as mentioned previously.
- Apply the ensemble Kalman filter with 500 ensemble members for 48 hours. Synthetic initial ensemble is created by adding zero-mean Gaussian noise to the reference initial condition with covariances set to the initial (diagonal) background error covariance matrix.
- Decorrelate the ensemble-based covariances using a decorrelation matrix ρ with decorrelation distance $L = 1000\,km$.
- Calculate $\mathbf{B}_0$ by averaging the covariances over the last 6 hours.

This method of creating a synthetic initial background error covariance matrix is totally empirical, but we found that the resulting background error covariance matrix performs well for several algorithms including 4D-Var. Moreover, the quality of the background error covariance matrix can be enhanced by the flow-dependent versions obtained by the used filters.

2.5.10 Results for shallow water model with linear observations

The assimilation time interval of the current experiment is 22 hours with observations available each hour. The number of burn-in steps in the Markov chain is set to 50. The size of the ensemble is chosen to be $\mathrm{N_{ens}} = 100$. Given the simple settings in this experiment, it is expected that Verlet (A.2) or any of the higher order symplectic integrators (A.3), (A.4), (A.5) will behave similarly with tuned parameters. Here we provide the results obtained using two-stage integrator with step size $h = 0.01$ and number of steps $m = 10$. The number of mixing steps is 5. Beginning with the second assimilation cycle, we noticed that the filter starts to show converge in a very small number of steps, typically 5 to 10 steps. Statistical tests of convergence should be considered to start the sampling process as soon as convergence is achieved and to avoid discarding states generated from the posterior.

We noticed that if the position Verlet is used instead, longer Hamiltonian trajectories are required while step sizes should be lowered in order to achieve both stability and fast space exploration. It is important to note that the quality of the background error covariance matrix and the way it is updated both have big impact on the performance of the filter. The flow-dependent background error covariance matrix is used to update the modeled background version. A hybrid version of the background error covariance matrix is created using (2.19) as a linear combination of the modeled version with the linear coefficient $\gamma = 0.5$. The forecast and analysis states obtained by EnKF and the sampling filter at



(a) Forecast: h (b) EnKF analysis: h (c) Sampling filter analysis: h

(d) Forecast: u (e) EnKF analysis: u (f) Sampling filter analysis: u

(g) Forecast: v (h) EnKF analysis: v (i) Sampling filter analysis: v

Figure 2.13: Data assimilation results for SWE on the sphere with linear observations where all components are observed. The plotted component of the state vector is indicated under each panel. The analysis shown is obtained after sequential assimilation of hourly observations for 22 hours. Only state at the last last assimilation cycle is plotted. The symplectic integrator used is the two-stage (A.3). The length of the Hamiltonian trajectory is $T = mh$, with $h = 0.01$, $m = 10$. The number of mixing steps is 5.

the last assimilation cycle are shown in Figure 2.13. The sampling filter results in an analysis state that is almost identical to the analysis obtained by EnKF. Figure 2.14 shows

the RMSE of the analysis states, over the 22 cycles, obtained by both EnKF and the HMC sampling filter. Clearly, in the presence of linear observation operator, the sampling filter competes with EnKF which is known to be optimal given the current settings. It is clear from both Figures 2.14, and 2.13 that the HMC is capable of handling high-dimensional models as well as EnKF. Despite the additional cost, the performance of the sampling filter under the current settings encourage us to test the sampling filter with larger models under more challenging settings. A comparison between EnKF, MLEF, and PF applied to SWE on the sphere, in the presence of both linear and nonlinear observation operators, can be found in [45].

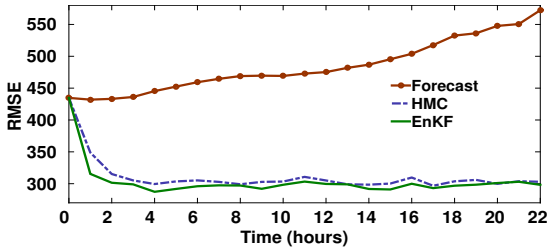
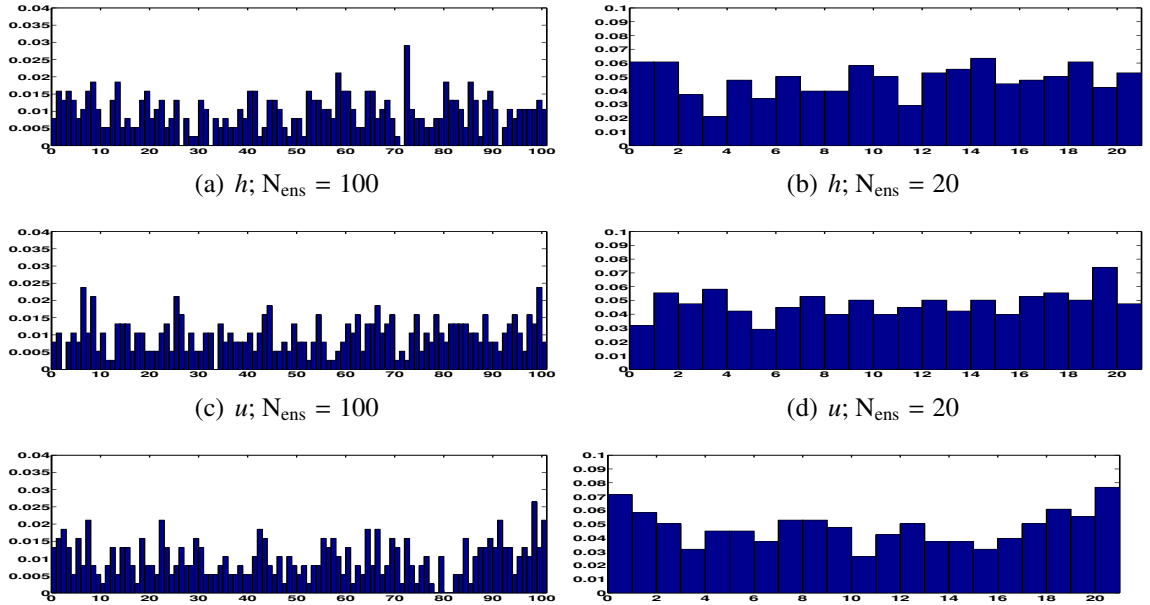


Figure 2.14: Data assimilation results for SWE on the sphere with linear observations where all components are observed. RMSE results of the sampling filter and EnKF are plotted against the Forecast(no assimilation) results. The symplectic integrator used is the two-stage (A.3). The length of the Hamiltonian trajectory is $T = 0.1$, with $h = 0.01$, $m = 10$. The number of mixing steps is 5.

Rank histograms of the uncorrelated components of the state vector along the 22 assimilation cycles are given in Figure 2.15. The first column of panels shows rank histograms where the ensemble size is $N_{ens} = 100$, while in the second column of panels the size of the ensemble is reduced to $N_{ens} = 20$. We see that even with 20 ensemble members, the rank histograms are approximately uniform. This gives the sampling filter a chance to work with high dimensional model where only small number of ensemble members is affordable.



(a) h; $N_{ens} = 100$ (b) h; $N_{ens} = 20$

(c) u; $N_{ens} = 100$ (d) u; $N_{ens} = 20$

(e) v; $N_{ens} = 100$ (f) v; $N_{ens} = 20$

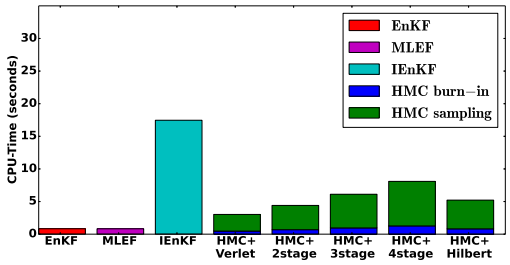
Figure 2.15: Data assimilation results for SWE on the sphere with linear observations where all components are observed. The rank histograms here consider only uncorrelated grid points of each of the three components. First row of panels 2.15(a), 2.15(c), 2.15(e), are obtained based on ensemble of 100 members. The following three panels 2.15(b), 2.15(d), 2.15(f) show rank histograms where the ensemble size is reduced to 20. The symplectic integrator used is the two-stage (A.3). The length of the Hamiltonian trajectory is $T = mh$, with $h = 0.01$, $m = 10$. The number of mixing steps is 5.

2.5.11 CPU-time usage

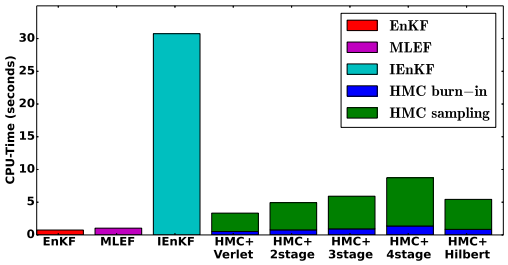
To complement the discussion provided in 2.4.1, we show the CPU usage, per assimilation cycle, of both the traditional filters and the HMC sampling filter.

The CPU-times for the experiments carried out using Lorenz-96 model are shown in Figure 2.16. As discussed in Section 2.4.1 the cost of the sampling filter depends on the settings of the Markov chain, and relies heavily on the choice of the parameters of the symplectic integrator of choice. Here we fix the parameters of all the symplectic integrators to $T = m \times h = 0.1$, with $m = 10$ and $h = 0.01$ (units) for both models. For experiments carried out using Lorenz-96 model, the size of the ensemble is 30 members,

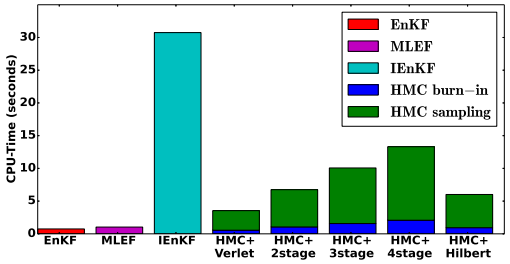
the number of burn-in steps is 50, and the number of mixing steps is 10. The blue parts of the bars report the CPU-time spent during the burn-in process, while the green parts show the time spent during sampling (collecting ensemble members after the Markov chain has reached stationarity). The reason behind the notable high CPU usage of IEnKF here is that we tune the IEnKF tolerance iteratively so that we guarantee the best possible results against which we can compare the HMC sampling results. While the average CPU-time of HMC sampling filter with Verlet integrator is roughly 4.7 times the CPU-time reported by EnKF, the cost of the HMC sampling filter is proportional to the number of stages of the symplectic integrator. This additional computational cost is justifiable in cases where traditional methods fail, or when ensemble member replenishment is needed.



(a) Linear observation operator (2.21)



(b) Quadratic observation operator with a threshold (2.22)

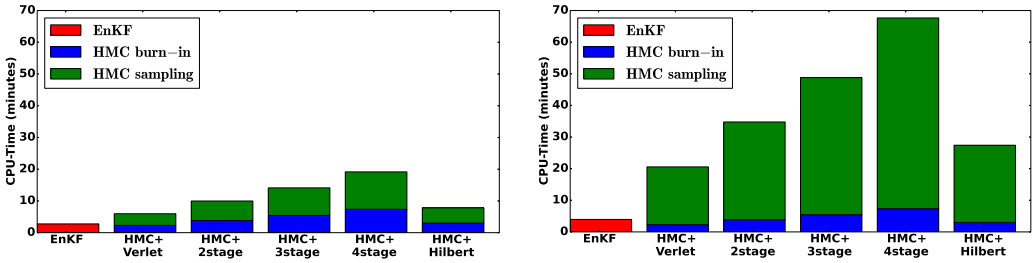


(c) Exponential observation operator (2.24)

Figure 2.16: CPU-time per assimilation cycle of DA with the Lorenz-96 model. The time reported is the average CPU-time taken over 100 identical runs of each experiment. The ensemble size is fixed to 30 members for all experiments here. The algorithm and the symplectic integrators used in the HMC sampler are both indicated under each panel. The length of the Hamiltonian trajectory is fixed to $T = mh$, with $h = 0.01$, $m = 10$. The number of burn-in steps and the number of mixing steps are 50 and 10, respectively.

Figure 2.17 shows the average CPU-time (in minutes) spent during each assimilation cycle

in the experiments carried out using the SWE model. For a linear observation operator we show the CPU-times for two cases with ensemble sizes of 100 and 20 members. The number of burn-in steps and the number of mixing steps are 50 and 5 respectively. With only 20 ensemble members collected the sampling filter with Verlet integrator takes approximately 2.16 longer than EnKF. As pointed out before, only the time spent during collecting sample points changes by varying the ensemble size, as seen by comparing results reported in Figures 2.17(b) and 2.17(a). To greatly reduce the time usage of the HMC sampler, one can run the burn-in process sequentially, to guarantee convergence to the right distribution, followed by a parallel run of the sampling process. The burn-in process itself can be replaced by a sub-optimal 3D-Var run to attain a local minimum of the negative-log of the posterior kernel.



(a) $N_{ens} = 20$ (b) $N_{ens} = 100$

Figure 2.17: Average CPU-time per assimilation cycle of the numerical experiments carried out using SWE model on a sphere. The time reported is the average CPU-time taken over 100 identical runs of each experiment. The ensemble size is indicated under each panel. The length of the Hamiltonian trajectory is fixed to $T = mh$, with $h = 0.01$, $m = 10$. The number of burn-in steps and the number of mixing steps are set to 50 and 5, respectively.

2.6 Conclusions and Future Work

This chapter proposes a sampling filter for data assimilation where the analysis scheme is replaced by sampling directly from the posterior distribution. A Hamiltonian Monte-Carlo technique is employed to generate a representative analysis ensemble at each time. The sampling filter avoids the need to develop tangent linear or adjoint models of the model solution operator. The sampling filter can work with highly nonlinear observation operators

and provides analysis ensembles that describe non-Gaussian posterior probability densities. The mean of the generated posterior ensemble provides the analysis (a minimum variance estimate of the state). The ensemble covariance offers an estimate of the analysis error covariance matrix and can be used to quantify the uncertainty associated with the analysis state. The implementation does not require the construction of full covariance matrices, which makes the method attractive for large scale data assimilation problems with operational models and complex observation operators.

Numerical experiments are carried out with the Lorenz-96 model with several observation operators with different levels of non-linearity and smoothness. The sampling filter provides results similar to EnKF for linear observations. For nonlinear observations the new filter outperforms EnKF, and its performance can be improved further with careful tuning of filter parameters. In addition, the sampling filter produces satisfactory results for test cases where EnKF, MLEF, and IEnKF fail.

Large scale ensemble filtering data assimilation problems are typically run on large parallel machines. One important challenge is the failure of subsets of nodes, which terminates some of the ensemble member runs, and leads to fewer ensemble members being available at the next time. Over several cycles the effective number of ensemble members can decrease considerably. The sampling strategy proposed herein can be used to replace dead ensemble members in any parallel implementation of the EnKF. In addition, the sampling filter can be used in combination with classical filters by building analysis ensembles that have members given by EnKF analysis (these members retain the history of the system) mixed with sampling members (which are consistent with the posterior probability density, but add new directions to explore and can therefore avoid filter divergence).

The computational performance of the sampling filter depends on tuning its parameters, especially the symplectic integration time step and the number of steps taken in the Markov chain between successive accepted ensemble members. Future work will focus on refining the strategies for parameter tuning in the context of large operational models at high-resolution, and on incorporating acceleration strategies for the HMC process. Use of non-Gaussian kernels such as Gaussian mixtures to represent the prior will also be investigated. A side by side comparison between the proposed sampling filter and the successful particle filters is planned. Balance is a concern for all filters including the sampling filter. A possible solution could be to restrict posterior PDF with physical information. This direction will be also investigated in future work.

Chapter 3

A Hybrid MonteCarlo Sampling Smoother for Four Dimensional Data Assimilation

3.1 Introduction

Data assimilation (DA) is the process of combining information from predictions made by imperfect models, from noisy observations, and from priors to produce a consistent description of the state of a dynamical system. The application of DA to large scale systems such as the atmosphere is of great practical interest. Two approaches for solving large DA problems have gained widespread acceptance. The first approach, originating from control theory, are variational methods such as the three-dimensional variational (3D-Var) and four-dimensional variational (4D-Var) strategies [12]. The variational methods find a maximum a-posteriori (MAP) estimate of the true state of the system. The second approach are the ensemble-based statistical estimation methods. The most successful family of ensemble data assimilation algorithms includes the ensemble Kalman filter (EnKF) [20] and its variants, the square-root Kalman filters [9], the ensemble adjustment Kalman filter [5], the ensemble transform Kalman filter [6], and efficient implementations of Kalman Filter, such as [67], using the Sherman-Morrison-Woodbury formula. All variants of EnKF provide a minimum variance estimate of the state by approximating the expected value of the posterior distribution. Variational and statistical estimation methods yield identical estimates (only) in the case of linear dynamics, linear observations, and Gaussian errors.

Both 3D-Var and EnKF are filtering methods that estimate the true state of the system at the specific time instances where observations are available. For many oceanographic, meteorological, and hydrological applications, it is advantageous to employ smoothing methods such as 4D-Var and EnKS that simultaneously use information from all observations available at different time points within an assimilation time window. Strong-constraint 4D-Var updates the state of the system at the initial time of an assimilation window given a background estimate of the initial condition and a set of observations distributed through this interval. The ensemble Kalman smoother, on the other hand, estimates the posterior distributions of the state at time points in the window given all past, present, and future observations (in the assimilation window).

4D-Var requires the derivation and implementation of the tangent linear and the adjoint numerical models, a challenging and effort-intensive task for large-scale models. The 4D-Var algorithm provides a single analysis state, the best posterior estimate of the state of the system. The uncertainty in the estimated state is not inherently provided by the 4D-Var algorithm [13]. Previous work proposed to quantify the uncertainty in the 4D-Var analysis, by approximating the analysis error covariance matrix using an ensemble of simulations [77, 78]. These schemes provide an approximation of the analysis error covariance matrix that is inconsistent with the 4D-Var analysis itself because the covariance estimates are usually obtained from independent schemes such as EnKF. Approaches to quantify 4D-Var analysis uncertainty based on subspace error decompositions [12, 13] are statistically consistent but require additional computational effort.

The EnKS is optimal when the observation operators are linear and the errors are Gaussian. However, these assumptions are unlikely to hold for real applications. The analysis ensemble generated by the EnKS allows to find a minimum variance estimate (e.g., ensemble mean), as well as a measure of the analysis uncertainty (e.g., the analysis error covariance matrix). When the posterior distribution is nearly Gaussian EnKS offers an efficient practical algorithm. However, when the observation operators are nonlinear and the errors are non-Gaussian, the EnKS is not expected to yield good results.

The Markov Chain Monte Carlo (MCMC) family of algorithms provides a powerful foundation to sample from complicated probability distributions. These algorithms work in general by generating a Markov chain whose stationary distribution is the target probability distribution. MCMC sampling is considered to be the gold standard [79] in data assimilation. The main practical limitation of MCMC is the considerable computational cost required to achieve convergence, and to explore the entire state space in the case of high dimensional state spaces. Scalable and accelerated MCMC algorithms are being continuously developed to improve convergence and space exploration. Hybrid Monte Carlo (HMC), also known as Hamiltonian Monte Carlo, is an accelerated MCMC sam-

pling algorithm that reduces the correlation between successive states by using Hamiltonian dynamics to generate proposal states [48, 80]. Moreover, HMC targets states with high acceptance probability leading to fast convergence and fast space exploration. To the best of our knowledge, HMC was first considered in the context of DA in [49] to solve a nonlinear inverse problem by minimizing the residual between the solution and ill-posed boundary conditions. Posterior error statistics are approximated by sampling the nearby states to the optimal state after convergence. In [49] a simulated annealing strategy is used during the sampling process where the minimum is obtained at a low temperature and posterior samples are collected at high temperatures. Annealing refers to the process of heating a solid material, and then cooling it slowly back to a solid state. The quality of the resulting product of the annealing process depends on the temperature profile used during the cooling stage. Simulated annealing (SA) [81, 82] is a mathematical analogy of the physical annealing process. SA is used to sample highly nonlinear functions in order to obtain a global minimum. Solving the weak-constraint 4D-Var problem using a gradient method then using HMC to estimate the analysis error statistics is also discussed in [4, Chapter 6].

This work develops a nonlinear non-Gaussian smoother to solve the four dimensional data assimilation problem. The new method uses a Hybrid Monte Carlo approach to sample the posterior distribution and is named the HMC smoother. This work extends the sampling filter, proposed in [36], to the four dimensional case where time-distributed observations are assimilated at once. HMC smoother provides an ensemble of states sampled from the posterior distribution of the state of the system at the initial time of the assimilation window. In a practical setting the smoothing step is carried out sequentially over consecutive assimilation windows. The generated ensemble encapsulates the uncertainty in the posterior distribution at the beginning of the current window; when propagated to the beginning of the next assimilation window it provides flow-dependent information about the background error distribution in the next assimilation cycle.

The use of HMC for solving smoothing problems was presented also in [50], where a generalized version of HMC is used in an attempt to reducing the number of chain steps required to ensure independence of the generated states. The underlying dynamical system in the generalized version of HMC is not Hamiltonian. There are several important differences between this work and [50]. In [50] the only source of uncertainty is model error, in form of additive random noise included in the dynamics. Here we work in the strong constraint 4D-Var framework and consider the model to be perfect; the system state is uncertain due to uncertainties in the initial conditions. While in [50] numerical experiments are carried out with a one-dimensional system, here we experiment with shallow water equations over the sphere, a moderately large multidimensional nonlinear model relevant

for geophysical applications. Finally, we propose to use higher order symplectic integrators, as tested in [36], to efficiently sample from complex posterior distributions that arise when the observation operators and models are highly nonlinear.

The remaining part of the chapter is organized as follows. Section 3.2 reviews the variational and the ensemble approaches for solving the data assimilation problem. The HMC smoother is presented in Section 3.3. Experimental settings and numerical results are discusses in Section 3.4. Conclusions and future directions are summarized in Section 3.5.

3.2 Data Assimilation

This section reviews the 4D-Var and the EnKS data assimilation schemes.

3.2.1 Four-dimensional variational data assimilation

4D-Var calculates the optimal initial condition for the state of the dynamical system over a specific assimilation time window, based on a background state and using all observations available within this time window [3]. The background initial state is usually the forecast produced by propagating the previous window analysis through the model dynamics. To be specific, let the current assimilation window be the time interval $[t_0\,, t_F]$. Given a background state $\mathbf{x}_0^{\mathrm{b}} = \mathbf{x}^{\mathrm{b}}[t_0]$, and a set of observations $\{\mathbf{y}_k = \mathbf{y}[t_k]\}_{k=0,1,\dots,m}$, available at the discrete time points $\{t_k\}_{k=0,1,\dots,m} \subset [t_0\,, t_F]$, the 4D-Var analysis is obtained by solving the following optimization problem:

$$\min_{\mathbf{x}_0} \mathcal{J}(\mathbf{x}_0) = \frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}}\right\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m} \|\mathcal{H}_k(\mathbf{x}_k) - \mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}}\,. \tag{3.1}$$

Here $\mathcal{H}_k$ is the observation operator (generally nonlinear) that maps the model space into the observation space at time point t_k. The dimension of observation space m is usually much lower than the dimension of the state space, that is $\mathrm{m} \ll \mathrm{N}_{\mathrm{state}}$. $\mathbf{B}_0$ is the background error covariance matrix, and $\mathbf{R}_k$'s are the observation error covariance matrices at each times t_k; $k = 1, \dots, m$. The background error covariance matrix determines how information from observed areas is extrapolated to unobserved regions or where observations are sparsely available [83]. The state $\mathbf{x}_k = \mathbf{x}[t_k]$, is produced by propagating the initial state $\mathbf{x}_0$ through the model dynamics from time t_0 to point t_k

$$\mathbf{x}_k = \mathcal{M}_{0,k}(\mathbf{x}_0). \tag{3.2}$$

The model solution operator $\mathcal{M}$ represents the discretized partial differential equations that govern the evolution of the dynamical system. Realistic atmospheric and ocean models typically have $\mathrm{N}_{\mathrm{state}} \sim 10^6 - 10^9$ state variables.

In this work we consider the strong-constraint 4D-Var case which assumes that the numerical model (3.2) is perfect. The methodology proposed here can be immediately extended to the weak-constraint 4D-Var framework [84, 85], where model errors are accounted for by adding the corresponding model error terms to equation (3.1) [83].

Perturbations (small errors $\delta\mathbf{x}$) of the state of the system evolve according to the tangent linear model:

$$\delta\mathbf{x}_k = \mathbf{M}_{0,k}(\mathbf{x}_0) \cdot \delta\mathbf{x}_0 \,, \quad t_0 \leq t \leq t_F \,, \tag{3.3}$$

where $\mathbf{M}_{0,k} = \partial\mathcal{M}_{0,k}/\partial\mathbf{x}[t_0]$ is the Jacobian of the model solution operator.

In strong-constraint 4D-Var the model dynamics act as constraints to the optimization problem (3.2). The optimal initial condition obtained by solving the optimization problem (3.1) constrained by the model dynamics (3.2), is referred to as the analysis state $\mathbf{x}_0^{\mathrm{a}} = \mathbf{x}^{\mathrm{a}}[t_0]$. The gradient of the cost functional (3.1) is:

$$\nabla_{\mathbf{x}_0}\mathcal{J}(\mathbf{x}_0) = \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}}) + \sum_{k=0}^{m} \mathbf{M}_{0,k}^T \mathbf{H}_k^T \mathbf{R}_k^{-1}(\mathcal{H}_k(\mathbf{x}_k) - \mathbf{y}_k) \,, \tag{3.4}$$

where $\mathbf{M}_{0,k}^T$ is the adjoint of the tangent linear model operator (3.3), $\mathbf{H}_k = \partial\mathcal{H}_k/\partial\mathbf{x}_k$ is the Jacobian of the observation operator $\mathcal{H}_k$, and $\mathbf{H}_k^T$ is its adjoint. Gradient-based minimization using (3.4) requires the development of the tangent linear and the adjoint models, which is a challenging task for high-dimensional complex models of practical interest. The performance of the optimization can be improved by using the second order derivative information and adaptive observations as described in [86, 87].

3.2.2 Bayesian interpretation of 4D-Var

The knowledge of the system state at the initial time t_0 prior to obtaining new observations is described by the background (prior) probability density $\mathcal{P}^b(\mathbf{x}_0)$. The "sampling model" gives the probability distribution of observations conditioned by the initial state $\mathcal{P}(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m|\mathbf{x}_0)$, under the belief that the dynamical model $\mathbf{x}_k = \mathcal{M}_{0,k}(\mathbf{x}_0)$ perfectly represents reality. From Bayes' theorem:

$$\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0) = \mathcal{P}(\mathbf{x}_0|\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m) \tag{3.5a}$$

$$= \frac{\mathcal{P}(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m|\mathbf{x}_0)\,\mathcal{P}^{\mathrm{b}}(\mathbf{x}_0)}{\mathcal{P}(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m)}, \tag{3.5b}$$

The posterior (analysis) probability distribution function (PDF) $\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)$ is the probability distribution of the initial state after incorporating the new knowledge contained in the observations. The denominator in (3.5b) is the marginal density of the observations and acts as a normalization factor.

The background and observations errors are usually assumed to have Gaussian distributions:

$$\mathcal{P}^{\mathrm{b}}(\mathbf{x}_0) \propto \exp\left(-\frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}}\right\|^2_{\mathbf{B}_0^{-1}}\right), \tag{3.6a}$$

$$\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k) \propto \exp\left(-\frac{1}{2}\left\|\mathcal{H}_k(\mathbf{x}_k) - \mathbf{y}_k\right\|^2_{\mathbf{R}_k^{-1}}\right), \tag{3.6b}$$

where $\mathbf{B}_0$ is the background error covariance matrix and $\mathbf{R}_k$'s are the observation error covariance matrices at times t_k; $k = 1, \ldots, m$. If the observation errors at different time points are independent, and the model is perfect, the joint sampling model can be written as

$$\mathcal{P}(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_m|\mathbf{x}_0) \propto \exp\left(\sum_{k=0}^{m}\left(-\frac{1}{2}\left\|\mathcal{H}_k(\mathbf{x}_k) - \mathbf{y}_k\right\|^2_{\mathbf{R}_k^{-1}}\right)\right). \tag{3.6c}$$

Bayes' rule (3.5) yields the following posterior PDF

$$\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0) \propto \exp\left(-\mathcal{J}(\mathbf{x}_0)\right), \tag{3.7a}$$

$$\mathcal{J}(\mathbf{x}_0) = \frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}}\right\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m}\left\|\mathcal{H}_k(\mathbf{x}_k) - \mathbf{y}_k\right\|^2_{\mathbf{R}_k^{-1}}. \tag{3.7b}$$

For nonlinear models and nonlinear observation operators the posterior (3.7) is not Gaussian. The kernel of the posterior is $\exp(-\mathcal{J}(\mathbf{x}_0))$, where $\mathcal{J}(\mathbf{x}_0)$ is the cost functional of the 4D-Var problem. 4D-Var computes the analysis $\mathbf{x}_0^{\mathrm{a}}$ as the minimizer of $\mathcal{J}$. The 4D-Var solution is the MAP estimate of the initial state under the assumptions that the background and observation errors are Gaussian. For highly nonlinear observation operators and highly nonlinear models the posterior can have multiple modes. In this case the 4D-Var numerical solution can be trapped in a local minimum of the cost functional.

The posterior distribution (3.5) contains the complete characterization of the uncertain initial state of the dynamical system. However, calculating the full posterior with high dimensional models is infeasible in practice. A practical approach is to describe the posterior probability density by an ensemble of states, and to use it to estimate moments of the distribution. Sampling directly from the posterior PDF of the initial condition (3.7) acts as a smoother; the ensemble mean provides an estimate of the true state of the system, and the ensemble covariance an estimate of the posterior uncertainty that is totally consistent with

the analysis state. In contrast, the current practice to use the analysis obtained from 4D-Var and the covariance obtained from EnKF or EnKS leads to inconsistent representations of uncertainty.

3.2.3 Ensemble Kalman filter and smoother

Filtering is the process of calculating the posterior distribution of the uncertain state of a dynamical system at a specific time given observations only available at that time instance. The ensemble Kalman filter [20] represents probability distributions by samples. Let $\{\mathbf{x}_k^{\mathrm{b}}(e)\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$ be an ensemble of forecast states at time t_k, and $\mathbf{y}_k = \mathbf{y}[t_k]$ the observation vector (set of measurements) at t_k. If the forecast (background) ensemble is represented by the matrix $\mathbf{X}_k^{\mathrm{b}}$, whose columns are the forecast ensemble members, then the updated (analysis) ensemble matrix $\mathbf{X}_k^{\mathrm{a}}$ at the same time t_k is obtained as [18]

$$\mathbf{X}_k^{\mathrm{a}} = \mathbf{X}_k^{\mathrm{b}} \cdot \mathbf{T}_k \,, \tag{3.8}$$

where the matrix $\mathbf{T}_k \in \mathbb{R}^{\mathrm{N_{ens}} \times \mathrm{N_{ens}}}$ is a nonlinear transformation constructed from the forecast ensemble and the observations at time t_k [18]. Square-root filters [9], the ensemble transform Kalman filter [6], and the ensemble adjustment Kalman filter [5] can all be written in the form (3.8) for specific choices of the transformation $\mathbf{T}_k$.

Smoothing is the process of calculating the posterior distribution of the uncertain states of a dynamical system given past, present, and future observations [88]. There are three approaches to smoothing: fixed-point, fixed-interval, and fixed-lag smoothing [88], with the second and third ones being the most popular [89]. Any scheme that can be used to solve any of the three smoothing problems can also be employed as a single fixed-interval smoothing scheme [88]. The ensemble smoother (ES) was introduced in [90] as a linear variance minimization algorithm. The ensemble Kalman smoother (EnKS) [91] employs an ensemble of states to describe distributions and obtains the posterior using the Kalman filter updates equations. EnKS is optimal in case of linear dynamics, Gaussian errors, and large number of ensemble members [4].

To construct EnKS [91] the EnKF update equations (3.8) are used repeatedly to develop a fixed-lag, and a fixed-interval smoothing algorithms. A fixed-point smoother can be written as [18]

$$\mathbf{X}_0^{\mathrm{s}} = \mathbf{X}_0^{\mathrm{a}} \cdot \prod_{k=0}^{m} \mathbf{T}_k \,. \tag{3.9}$$

The update equation (3.9) is used recursively for fixed-interval smoothing, where smoothed ensembles are obtained at specified set of times, and they are conditioned only on obser-

vations available at later times in the interval. Ravela and McLaughlin [89] presented efficient, fast versions of the fixed-interval and the fixed-lag EnKS. The fast fixed-interval smoother has a computational cost that scales linearly with respect to the length of the interval. In this work, we use the fast fixed-interval EnKS [89], with a single smoothing point (fixed-point smoother) chosen at the beginning of the time interval.

EnKS computes the minimum variance estimate of the state. This is not expected to be very accurate if the observations are highly nonlinear or if the Gaussianity assumptions are severely violated. As shown in Section 3.4, the HMC sampling smoother proposed herein is capable of handling nonlinear observation and model operators, and consequently produces posterior estimates that are more useful than the EnKS ensemble, and contain more information than the 4D-Var MAP analysis. Hybrid methods such as [92] make use of optimization over ensembles using the trust-region framework.

3.3 The hybrid Monte-Carlo sampling smoother

The most popular and successful class of sampling algorithms is the Markov chain Monte-Carlo (MCMC) [57], first introduced by Metropolis *et al.* [58]. MCMC algorithms sample from a general probability distribution $\mathcal{P}(\mathbf{x})$ by building a Markov chain whose invariant distribution is $\mathcal{P}(\mathbf{x})$. MCMC algorithms have an advantage of not requiring the normalization of target distributions. However, traditional MCMC samplers are often considered impractical for large dimensional problems due to the following drawbacks: the Markov chain may take a very long time to reach stationarity. A large number of (burn-in) states are generated and discarded before starting the sampling process, in order to guarantee that the collected samples are obtained from the true target PDF. The samples should be independent, however the Markov chain is not completely memoryless; in order to achieve independence of sampled states, the sampler usually drops some intermediate states between each selected state. Another drawback of most of Monte-Carlo sampling methods is the curse of dimensionality [57]: as the dimension of the state spaces grows, the number of sample members needed to represent the probability distribution, grows rapidly. The number of samples required to efficiently represent the probability distribution can be controlled if the sampler surveys sufficiently fast the entire state space. The sampler can become trapped in a high-probability mode of a multi-modal distribution, and fail to represent the other probability modes.

3.3.1 Hybrid Monte-Carlo

Hybrid/Hamiltonian Monte-Carlo (HMC) [48] follows an auxiliary-variable approach in order to alleviate the limitations of the traditional MCMC algorithms.

The phase space of a Hamiltonian dynamical system consists of points $(\mathbf{p}, \mathbf{x}) \in \mathbb{R}^{2\,\mathrm{N_{state}}}$, where $\mathbf{x} \in \mathbb{R}^{\mathrm{N_{state}}}$ is the position variable, and $\mathbf{p} \in \mathbb{R}^{\mathrm{N_{state}}}$ is the momentum variable. The Hamiltonian dynamics is governed by the set of ordinary differential equations (ODEs):

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \nabla_{\mathbf{p}} H(\mathbf{p}, \mathbf{x}), \\ \frac{d\mathbf{p}}{dt} &= -\nabla_{\mathbf{x}} H(\mathbf{p}, \mathbf{x}), \end{aligned} \tag{3.10a}$$

where the Hamiltonian function H describes the total energy of the system

$$H(\mathbf{p}, \mathbf{x}) = \frac{1}{2}\,\mathbf{p}^T \mathbf{M}^{-1} \mathbf{p} + \mathcal{J}(\mathbf{x}). \tag{3.10b}$$

The first term of the Hamiltonian (3.10b) represents the potential energy of the system, while the second term corresponds to the kinetic energy. The exact (analytic) flow of the Hamiltonian system (3.10a) advances the solution in time from $t = 0$ to $t = T$:

$$\Phi_T : \mathbb{R}^{2\mathrm{N_{state}}} \to \mathbb{R}^{2\mathrm{N_{state}}}; \quad \Phi_T\big(\mathbf{p}[0], \mathbf{x}[0]\big) = \big(\mathbf{p}[T], \mathbf{x}[T]\big). \tag{3.11}$$

This flow cannot be calculated exactly in practice, and has to be approximated by an equivalent numerical solution using a time reversible symplectic integrator [62, 61]. The most common symplectic integrator is leapfrog (Störmer–Verlet) [62, 61]. One step of the position Verlet algorithm advances the solution of the Hamiltonian equations (3.10a) from time t_j to time $t_{j+1} = t_j + h$ using:

$$\mathbf{x}_{j+1/2} = \mathbf{x}_j + \frac{h}{2}\,\mathbf{M}^{-1}\,\mathbf{p}_j, \tag{3.12a}$$

$$\mathbf{p}_{j+1} = \mathbf{p}_j - h\,\nabla_{\mathbf{x}} \mathcal{J}(\mathbf{x}_{j+1/2}), \tag{3.12b}$$

$$\mathbf{x}_{j+1} = \mathbf{x}_{j+1/2} + \frac{h}{2}\,\mathbf{M}^{-1}\,\mathbf{p}_{j+1}. \tag{3.12c}$$

The optimal time step size h must satisfy $h \propto (1/\mathrm{N_{state}})^{1/4}$ [71], and careful empirical tuning of the step size is usually required for good performance [36]. Several other symplectic integrators with more stages and higher accuracy than Verlet have also been developed [63]. An infinite dimensional time integrator was also introduced in [64].

For practical considerations it is advisable to split the interval $[0, T]$ where the Hamiltonian system evolves into m smaller sub steps of length $h = T/m$. The flow of the numerical solution obtained by the symplectic integrator will be denoted by $\tilde{\Phi}_T$ and is an approximation of the exact flow Φ_T.

The key idea in HMC sampling is to add an auxiliary variable $\mathbf{p}$ to the target variable $\mathbf{x}$ and sample from the joint probability distribution of $(\mathbf{x}, \mathbf{p})$. The auxiliary variable is chosen such that the sampling procedure from the joint distribution is much faster than sampling from the marginal distribution of the target variable. In HMC sampling the target and the auxiliary variables are thought of as the position and momentum components of a Hamiltonian system, respectively. The Hamiltonian dynamics of the system serves as a transition kernel to the Markov chain.

The kernel of the stationary probability distribution of the Hamiltonian system (3.10) is [57]

$$\exp\left(-H(\mathbf{p}, \mathbf{x})\right) = \exp\left(-\frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} - \mathcal{J}(\mathbf{x})\right) \tag{3.13a}$$

$$= \exp\left(-\frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}\right) \cdot \pi(\mathbf{x})\,, \tag{3.13b}$$

where $\pi(\mathbf{x}) = \exp\left(-\mathcal{J}(\mathbf{x})\right)$ is the probability distribution of the position variable. The joint probability distribution of the state $(\mathbf{p}, \mathbf{x})$ in the phase space $\mathbb{R}^{2\,\mathrm{N_{state}}}$ is the product of the marginal distributions of both the position and the momentum. This simply means that the two variables $\mathbf{x}$ and $\mathbf{p}$ are independent [62]. Independence of both position and momentum makes it possible to sample from the marginal distribution of each variable by sampling from their joint distribution. The marginal PDF of the momentum variable is a Gaussian distribution with zero mean and covariance matrix $\mathbf{M}$ (also known as the mass matrix), i.e., $\mathbf{p} \sim \mathcal{N}(0, \mathbf{M})$.

Let $\mathbf{x} \sim \pi(\mathbf{x})$ be a random variable that is the target of the MCMC sampling algorithm. View $\mathbf{x}$ as the position variable in the Hamiltonian system, and add the momentum $\mathbf{p}$ as an auxiliary variable. The symplectic integrator is used to propose a state that is either accepted or rejected using an acceptance/rejection rule based on the loss of energy. Algorithm 3 [62] summarizes the HMC steps to sample from the probability distribution $\pi(\mathbf{x})$. The loss of energy between the current and the proposed state is usually calculated as the difference between the Hamiltonians at the current and the proposed states:

$$\Delta H = H(\mathbf{p}^*, \mathbf{x}^*) - H(\mathbf{p}_k, \mathbf{x}_k)\,. \tag{3.16}$$

This equation (3.16) is valid for the Verlet (3.12), two-stage, three-stage, and four-stage symplectic integrators [62, 63]. See [36] for details on different symplectic integrators and

Algorithm 3 HMCMC Sampling [62].

1: Initialize the Markov chain. Preferably $(\mathbf{p}_0, \mathbf{x}_0)$ should have high probability w.r.t. the target distribution.

2: At each step k of the Markov chain draw the random auxiliary variable $\mathbf{p}_k \sim \mathcal{N}(0, \mathbf{M})$.

3: Use a symplectic numerical integrator (e.g. position Verlet) to advance the current state $(\mathbf{p}_k, \mathbf{x}_k)$ by a time increment T to obtain a *proposal* state $(\mathbf{p}^*, \mathbf{x}^*)$:

$$(\mathbf{p}^*, \mathbf{x}^*) = \tilde{\Phi}_T\big((\mathbf{p}_k, \mathbf{x}_k)\big). \tag{3.14}$$

4: Use the Hamiltonian (3.10b) to approximate the loss of energy ΔH.

5: Calculate the acceptance probability:

$$a^{(k)} = 1 \wedge e^{-\Delta H}. \tag{3.15}$$

6: Discard both $\mathbf{p}^*$ and $\mathbf{p}_k$.

7: *(Acceptance/Rejection)* Draw a uniform random variable $u^{(k)} \sim \mathcal{U}(0, 1)$:

i- If $a^{(k)} > u^{(k)}$ accept the proposal as the next sample: $\mathbf{x}_{k+1} := \mathbf{x}^*$;

ii- If $a^{(k)} \leq u^{(k)}$ reject the proposal and continue with the current state: $\mathbf{x}_{k+1} := \mathbf{x}_k$.

8: Repeat steps 2 to 7 until sufficiently many distinct samples are drawn.

corresponding expressions for energy. The length of the Hamiltonian trajectory T and the number of steps m are parameters to be tuned by the user [60]. The step size h can be chosen empirically such as to achieve a rejection rate that falls between 25% and 30%, which is the approach we followed in our settings, and the number of steps m should be generally large to reach far points in the state space [60]. Another user-tunable parameter is the mass matrix $\mathbf{M}$, a symmetric positive definite matrix that represents the covariance of the momentum variable. The choice of the mass matrix does not alter the fact that HMC sampling Algorithm 3 converges to the stationary distribution $\pi(\mathbf{x})$. However, a good choice of $\mathbf{M}$ can considerably improve sampling efficiency. One popular and simple choice is to take $\mathbf{M}$ a constant multiple of the identity matrix. Ideally, if the variances of the target distribution $\pi(\mathbf{x})$ are known (or can be approximated), the diagonal of $\mathbf{M}$ should be chosen as the corresponding precisions (reciprocals of these variances) [60]. We found that this choice results in a very fast convergence of the chain to stationarity.

HMC sampling Algorithm 3 tends to explore the state space faster than traditional MCMC, and the acceptance probability of all generated states is close to one. Several enhancements to the HMC sampling, such as parallel tempering [93, 94], have been proposed to guarantee that the algorithm escapes local modes of high probability. The idea of parallel tempering is to run several replicas of the HMC sampler initialized randomly at different temperatures. The chain running at high temperature samples large volumes of the phase space, while low temperature chains sample local regions of the phase space. While low temperature chains can be entrapped in local minima, parallel tempering allows chains running at different temperatures to exchange complete configurations. This information exchange process helps low temperature chains to escape local minima and consequently increases the performance of the sampler.

3.3.2 Sampling smoother algorithm

We now present the HMC smoother (smoothing by sampling) that simultaneously accounts for all observations available within a specific assimilation window to obtain posterior estimates of the initial system state.

Consider the assimilation window $[t_0, t_F]$ with a set of observations available at times $t_0, t_1, \ldots, t_m$ inside the window, where $t_m \equiv t_F$. Under the assumptions discussed in Section 3.2.2 the posterior (analysis) probability distribution of the initial state $\mathbf{x}_0$ takes the form (3.7). We seek to sample from this posterior distribution using the HMC approach. For this we set the potential energy term in (3.10b) to be the 4D-Var cost functional (3.7b). Consequently the target probability distribution $\pi(\mathbf{x})$ coincides with the 4D-Var posterior

distribution (3.7), i.e., $\pi(\mathbf{x}) = \exp(-\mathcal{J}(\mathbf{x}))$.

The smoother works sequentially *over consecutive assimilation windows* by applying the forecast and analysis (sampling) steps in succession. In the forecast step each state of the analysis ensemble is propagated in time to the end of the previous assimilation window (the beginning of the current window). The result of the forecast step is a forecast ensemble $\mathbf{X}^{\mathrm{b}}$ (or just a single background state $\mathbf{x}_0^{\mathrm{b}}$) at the beginning of the current time window, i.e. at t_0. One can just propagate the analysis state (e.g. the mean of the analysis ensemble) to obtain the current background state $\mathbf{x}_0^{\mathrm{b}}$. However, propagating the full analysis ensemble makes it possible to build an ensemble-based (flow-dependent) background error covariance matrix at the beginning of the current window. This background error covariance matrix includes the errors of the day, and can considerably enhance the quality of the analyses generated by a data assimilation scheme. We will assume herein that the full forecast ensemble is generated in the forecast step. In the analysis step, the HMC sampling strategy summarized in Algorithm 3 is applied to obtain the analysis ensemble at the initial time of the current assimilation window.

The HMC sampling smoother is detailed in Algorithm 4.

Algorithm 4 The Proposed Sampling Smoother

1: **Analysis step:** Given the background state and observations, draw an ensemble of initial states from the posterior distribution (3.7) as follows:

   i- Calculate an ensemble-based forecast error covariance matrix $\mathbf{B}_0^{\text{ens}}$, and use it together with a fixed (modeled) matrix to construct the background error covariance matrix $\mathbf{B}_0$ [36, 37]. (This step can be omitted by using the modeled background error covariance matrix; however, the use of forecast ensemble is expected to improve the quality of the generated analysis ensemble.)

   ii- Build the mass matrix $\mathbf{M}$ as a diagonal matrix such that $\text{diag}\,(\mathbf{M}) = \text{diag}\left(\mathbf{B_0^{-1}}\right)$.

   iii- Initialize the Markov chain to the best estimate of the current state available, e.g., the background state $\mathbf{x}_0^{\text{b}}$, or a sub-optimal 4D-Var solution. A good choice speeds up the convergence of the chain.

   iv- Follow the steps in Algorithm 3 to generate the chain and select ensemble members after the chain reaches stationarity. Dropping a small number (5 to 10) steps between each selected states helps to ensure the independence of the generated ensemble members.

2: **Forecast step:** Propagate each member of the analysis ensemble, using the full forward model, to the end of the current assimilation window (beginning the next assimilation window).

The generated ensemble of states $\{\mathbf{x}_0^{\text{a}}(e)\}_{e=1,2,\ldots,\text{N}_{\text{ens}}}$, samples the posterior PDF $\mathcal{P}^{\text{a}}(\mathbf{x}_0)$, and can be used to calculate the best estimate of the initial condition of the system (e.g., the mean ($\overline{\mathbf{x}}_0^{\text{a}}$) of the ensemble), and to estimate the analysis error covariance matrix $\mathbf{A}_0$:

$$\overline{\mathbf{x}}_0^{\text{a}} = (\text{N}_{\text{ens}})^{-1} \sum_{e=1}^{\text{N}_{\text{ens}}} \mathbf{x}_0^{\text{a}}(e)\,, \tag{3.17a}$$

$$\Delta\mathbf{X}_0^{\text{a}} = [\mathbf{x}_0^{\text{a}}(1) - \overline{\mathbf{x}}_0^{\text{a}}, \ldots, \mathbf{x}_0^{\text{a}}(\text{N}_{\text{ens}}) - \overline{\mathbf{x}}_0^{\text{a}}]\,,$$

$$\mathbf{A}_0 = (\text{N}_{\text{ens}} - 1)^{-1} \left(\Delta\mathbf{X}_0^{\text{a}} \left(\Delta\mathbf{X}_0^{\text{a}}\right)^T\right)\,. \tag{3.17b}$$

The forecast and analysis steps are repeated sequentially on subsequent assimilation windows. The propagated ensemble can be used to estimate the analysis covariance at the final time using (3.17b), such as to provide "flow-dependent" information for the background error covariance matrix used in the subsequent assimilation interval.

3.4 Numerical Experiments

The proposed HMC sampling smoother is tested against the EnKS and the 4D-Var schemes on two numerical models. We first illustrate the distinctive features of the HMC smoother with a simple one-dimensional model with a nonlinear observation operator and a bimodal posterior distribution. Next, we employ the shallow water on the sphere to test the sampling smoother on a problem relevant to geophysics, and compare its performance with the conventional EnKS, and 4D-Var schemes.

3.4.1 A one-dimensional model

Consider the following model:

$$\frac{d\mathbf{x}}{dt} = -\frac{dV(\mathbf{x})}{d\mathbf{x}}\,, \tag{3.18a}$$

$$V(\mathbf{x}) = (\mathbf{x}+1)^2\,(\mathbf{x}-1)^2\,, \tag{3.18b}$$

that describes the position of a particle over the entire real line moving under the effect of the potential field (3.18b). This model is similar to the one used in [50]. The potential field has two local minima at ± 1, which are expected to act as attractors for the particle. We set the reference initial condition to $\mathbf{x}_0^{\text{true}} = -0.15$, and chose the background initial condition to be $\mathbf{x}_0^{\text{b}} = 0.1$. Note that the true and the background initial conditions lie in the basins of attraction of different equilibria. The background errors are assumed to be normally distributed with zero mean and standard deviation $\sigma_{\mathbf{x}_0} = \sqrt{2}$.

Synthetic observations are obtained from the reference solution by applying the quadratic observation operator

$$\mathcal{H}(\mathbf{x}_k) = (\mathbf{x}_k)^2\,. \tag{3.19}$$

Observation errors are assumed to be Gaussian with zero mean and standard deviation $\sigma_{\text{obs}} = 0.05$. The simulation time window is $[t_0, t_F] = [0, 0.12]$ (units), with equally spaced 12 observation points. The posterior distribution of the initial state reads

$$\mathcal{P}^{\text{a}}(\mathbf{x}_0) \propto \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_0 - 0.1}{1.41}\right)^2 - \frac{1}{2}\sum_{k=1}^{m=12}\left(\frac{\mathbf{x}_k^2 - \mathbf{y}_k}{0.05}\right)^2\right)\,, \tag{3.20}$$

where $\mathbf{x}_k$ is obtained by propagating $\mathbf{x}_0$ forward in time, from $t_0 = 0$ to $t_k = k \times 0.01$ (units), using the model (3.18). The non-normalized posterior density (3.20) is illustrated in Figure 3.1. Traditional assimilation methods, like 4D-Var and EnKS, are expected to

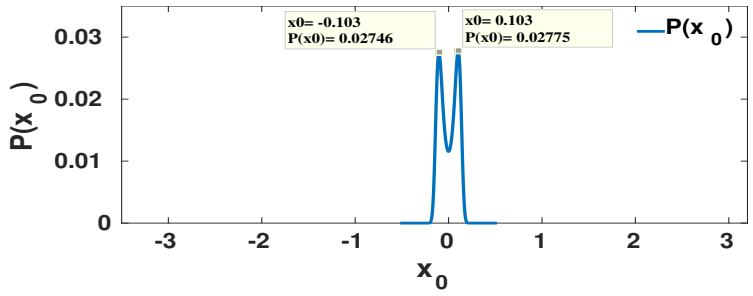
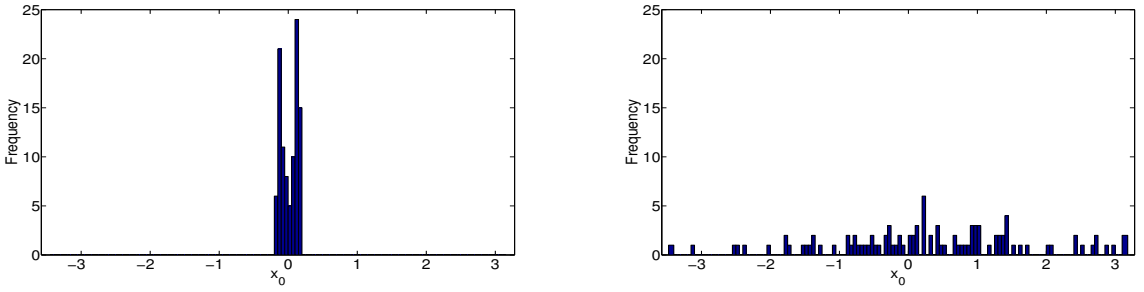


Figure 3.1: The non-normalized kernel of the posterior distribution (3.20). The right peak is slightly higher than than the left one as a result of the prior being a Gaussian distribution centered around $\mathbf{x}^{\mathrm{b}} = 0.1$ with small standard deviation ($\sigma_{\mathbf{x}_0} = \sqrt{2}$) Given the current settings, the right peak occurs at $\mathbf{x}_0 = 0.103$ with $P(\mathbf{x}_0) = 0.02775$ while the left peak occurs at $\mathbf{x}_0 = -0.103$ with $P(\mathbf{x}_0) = 0.02746$.

have difficulties capturing the bimodal nature of the posterior distribution of the initial condition. Since the prior PDF is a Gaussian centered around the background state $\mathbf{x}^{\mathrm{b}} = 0.1$ with standard deviation $\sigma_{\mathbf{x}_0} = \sqrt{2}$, the right peak in Figure 3.1 is slightly taller than the left peak. With Gaussian background prior centered around one of the peaks, smaller standard deviation would damp the other peak. Capturing only that right peak completely misses the true solution, which is negative. Numerical results presented below show that the proposed HMC smoother is capable of building a representative ensemble from the bimodal posterior distribution. The analysis ensemble can then be used to draw more useful conclusions (e.g. statistics) than what can be obtained from analysis results obtained by the traditional methods.

Numerical results with the one-dimensional model

HMC smoothing was carried out to collect an ensemble of 100 members from the posterior (3.20). We tested several symplectic integrators [36], and found that all show similar behavior. We chose the position Verlet symplectic integrator due to its minimal computational cost for all our experiments. The Hamiltonian system step size is empirically tuned to $T = 0.1$, with step length $h = 0.01$, and number of steps $m = 10$. The number of burn-in steps is chosen to be 20 (for this simple model we already know that the forecast state 0.1 lies in the support of the posterior and the burn-in steps could be omitted; in general one can incorporate convergence tests to shorten the number of burn-in steps and ensure that the collected samples are from the target distribution). Four states are dropped between consecutive selected states at stationarity to guarantee the independence of the samples.



(a) HMC sampling smoother. (b) Ensemble Kalman smoother.

Figure 3.2: Histograms of the analysis ensembles generated by HMC smoother, and EnKS. The number of ensemble members generated by each smoother is 100. For the HMC smoother the step of the symplectic integrator (3.12) is $T = 0.1$, with $h = 0.01$ and $m = 10$.

The histograms of the analysis ensembles obtained with the HMC smoother and EnKS are shown in Figure 3.2. The HMC smoother generates an analysis ensemble that matches the kernel shown in Figure 3.1, but EnKS fails to generate an accurate analysis ensemble. The most likely state seems to be located in the correct place. A single analysis state (best estimate) in this case might be misleading. One needs to consider more than one analysis with certain probability to give better description of the true state of the system in case of multi-modal systems.

The 4D-Var algorithm is expected to be trapped in a local minimum of the posterior distribution. Since the background state is closer to $+1$ than to -1, and since the observations (3.19) are insensitive to the sign of solution, we expect the analysis to follow the behavior of the true solution but with the opposite sign. This is confirmed by results in Figure 3.3.

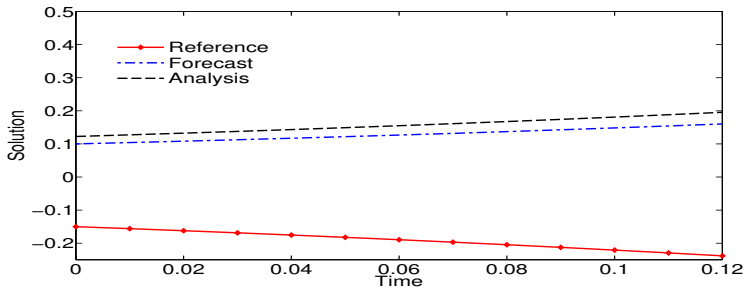


Figure 3.3: Data assimilation results using 4D-Var, together with the forecast and reference trajectories plotted over the assimilation window.

3.4.2 Shallow water model on the sphere

The shallow water equations have been used extensively as a simplified model of the atmosphere that contains the essential wave propagation mechanisms found in general circulation models (GCMs)[75]. The shallow water equations in spherical coordinates are [95]:

$$\frac{\partial u}{\partial t} + \frac{1}{a\cos\theta}\left(u\frac{\partial u}{\partial \lambda} + v\cos\theta\frac{\partial u}{\partial \theta}\right) - \left(f + \frac{u\tan\theta}{a}\right)v + \frac{g}{a\cos\theta}\frac{\partial h}{\partial \lambda} = 0\,, \tag{3.21a}$$

$$\frac{\partial v}{\partial t} + \frac{1}{a\cos\theta}\left(u\frac{\partial v}{\partial \lambda} + v\cos\theta\frac{\partial v}{\partial \theta}\right) + \left(f + \frac{u\tan\theta}{a}\right)u + \frac{g}{a}\frac{\partial h}{\partial \theta} = 0\,, \tag{3.21b}$$

$$\frac{\partial h}{\partial t} + \frac{1}{a\cos\theta}\left(\frac{\partial (hu)}{\partial \lambda} + \frac{\partial (hv\cos\theta)}{\partial \theta}\right) = 0\,. \tag{3.21c}$$

Here f is the Coriolis parameter, given by $f = 2\Omega\sin\theta$, where Ω is the angular speed of the rotation of the Earth. In addition, h represents the height of the homogeneous atmosphere, u and v are the zonal and meridional wind components, respectively. The latitudinal and longitudinal directions are respectively denoted by θ and λ. The radius of the Earth is denoted by a and g is the acceleration due to gravity. The space discretization is performed using the unstaggered Turkel-Zwas scheme [96]. The discretization has nlon=72 nodes in longitudinal direction and nlat=36 nodes in the latitudinal direction. The semi-discretization in space leads to a system of ordinary differential equations:

$$\mathbf{x}' = f(t,\mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_0; \quad t_0 = 0,\ t_F = 9\,\text{(hours)}. \tag{3.22}$$

The vector $\mathbf{x} \in \mathbb{R}^n$ with $n = 3 \times \text{nlat} \times \text{nlon}$ contains discrete versions of the zonal wind, meridional wind, and the height variables. We perform the time integration using a 5^{th} order Runge-Kutta method. This time-integrator is part of the MATLODE suite [97], which also has sensitivity analysis capabilities.

Observations and background information

A reference initial condition is used to generate a reference trajectory. Synthetic observations are created from the reference trajectory by adding Gaussian noise with zero mean to the observed components. In the present work, we incorporate two observation operators;

1. a linear observation operator where all components of the state vector are observed, i.e.

$$\mathcal{H}_k(\mathbf{x}_k) = \mathbf{x}_k = (u_1, \ldots, u_{\text{N}_{\text{state}}/3}, v_1, \ldots, v_{\text{N}_{\text{state}}/3}, h_1, \ldots, h_{\text{N}_{\text{state}}/3})^T \in \mathbb{R}^{\text{N}_{\text{state}}}\,, \tag{3.23a}$$

2. a nonlinear observation operator where only height and wind magnitude are observed, that is:

$$\begin{aligned}\mathcal{H}_k(\mathbf{x}_k) &= \mathcal{H}\Big((u_1,\ldots,u_{\mathrm{N_{state}}/3},\, v_1,\,\ldots,v_{\mathrm{N_{state}}/3},\, h_1,\,\ldots,\,h_{\mathrm{N_{state}}/3})^T\Big)\\ &= \Big(\sqrt{u_1^2+v_1^2},\,\ldots,\,\sqrt{u_{\mathrm{N_{state}}/3}^2+v_{\mathrm{N_{state}}/3}^2},\, h_1,\,\ldots,\,h_{\mathrm{N_{state}}/3}\Big)^T \in \mathbb{R}^{\frac{2\,\mathrm{N_{state}}}{3}},\end{aligned} \tag{3.23b}$$

where in both (3.23a), and (3.23b), the time index k is dropped off the components for clarity.

In the first case, where the linear observation operator (3.23a) is used, the observation error standard deviation for the height component is set to 1.5% of the average magnitude of the reference height component in the reference trajectory. The observation error standard deviation for the wind components is set to 10% of the average magnitude of the reference wind component in the reference trajectory. The value of the observation error standard deviation of the height is 700, and the observation error standard deviation of each of the two wind components is 3.

In the case of the nonlinear observation operator (3.23b), the observation error standard deviations are set to 4.5, and 700 for the wind magnitude, and the height respectively.

The initial background state is created by perturbing the reference initial condition with a Gaussian error drawn from the distribution $\mathcal{N}(0, \mathbf{B}_0)$, with a modeled background error covariance matrix. The background error covariance $\mathbf{B}_0$ is modeled as follows:

- Start with a diagonal background error covariance matrix. The standard deviation of the background errors for the height component is 2% of the average magnitude of the reference height component in the reference initial condition. The standard deviation of the background errors for the wind components is 15% of the average magnitude of the reference wind component in the reference initial condition.
- Synthetic initial ensemble is created by adding zero-mean Gaussian noise to the reference initial condition with covariances set to the initial (diagonal) background error covariance matrix. Apply the ensemble Kalman filter for 48 cycles with observations obtained each hour. The uncertainties in observations are fixed to 1.5%, and 10% for the height and wind components respectively. The synthetic observations are obtained by adding Gaussian noise with zero mean and standard deviation equal to the uncertainty level multiplied by the average magnitude of the corresponding component (height and wind) in the initial condition.

- Decorrelate the ensemble-based covariances using a decorrelation matrix ρ with decorrelation distance $L = 1000\,km$.
- Calculate $\mathbf{B}_0$ by averaging the ensemble covariances over the last 6 hours with one matrix per hour. In this version the background noise levels are no longer 2% and 15%.

This method of creating a synthetic initial background error covariance matrix is empirical, but we found that the resulting background error covariance matrix performs well for several algorithms including 4D-Var. Enhancing the quality of this background error covariance matrix can be done by making use of the ensembles generated by the sampling smoother, an idea that we will investigate in future work.

Data assimilation experiments with this model were conducted for three consecutive assimilation windows. The time interval of the first assimilation window is $[0, 6]$ hours, the second window is $[6, 14]$ hours, and the third is $[14, 22]$ hours. The short first window can be regarded as a spin-off period for the data assimilation system. Hourly (synthetic) observations are available on each of the three windows, with a total of 6 observation times in the first window, and 8 observation times in each of the last two windows.

Two experiments were conducted. In the first one the background error covariance matrix $\mathbf{B}_0$ is kept fixed for each of the three windows. In the second experiment $\mathbf{B}_0$ is updated with information from the generated ensemble according to the following expression:

$$\mathbf{B}_0^{\text{hybrid}} = \gamma \times \mathbf{B}_0^{\text{modeled}} + (1 - \gamma) \times \mathbf{B}_0^{\text{ensemble}}\,, \tag{3.24}$$

where $\mathbf{B}_0^{\text{hybrid}}$ is the updated version of $\mathbf{B}_0$, and $\mathbf{B}_0^{\text{modeled}}$ is the fixed version used in the first experiment. The scalar weight γ is a number in the interval $[0, 1]$. Selecting $\gamma = 1$ ignores the error-of-the-day, while $\gamma = 0$ forces the use of only the flow-dependent background error covariance matrix obtained from the ensemble, possibly leading to a singular covariance matrix. In our experiments we chose $\gamma = 0.75$.

The error metric used to compare analyses against the reference solution is the root mean squared error (RMSE):

$$\mathbf{RMSE} = \sqrt{\frac{1}{\mathrm{N}_{\text{state}}} \sum_{i=1}^{\mathrm{N}_{\text{state}}} (\mathbf{x}_i - \mathbf{x}_i^{\text{true}})^2}\,, \tag{3.25}$$

where $\mathbf{x}^{\text{true}}$ is the reference state of the system. The RMSE is calculated hourly along the trajectory over each assimilation window.

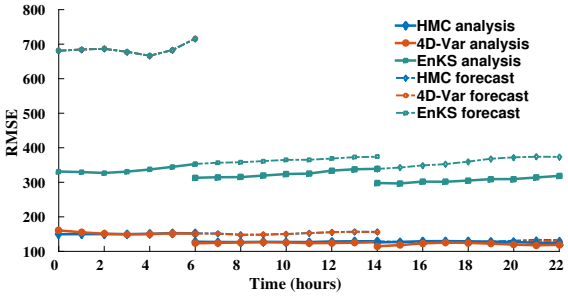
Numerical results with the shallow water on the sphere model

The numerical optimization step in 4D-Var is carried out using the the LBFGS routine implemented in the Poblano optimization toolbox [98]. Here the optimization process is stopped when the norm of the gradient is $1e-10$ or when the relative function tolerance hits $1e-6$. The optimization process takes at least 45 iterations of LBFGS to converge for the experiment considered here; (see Table 3.1) .

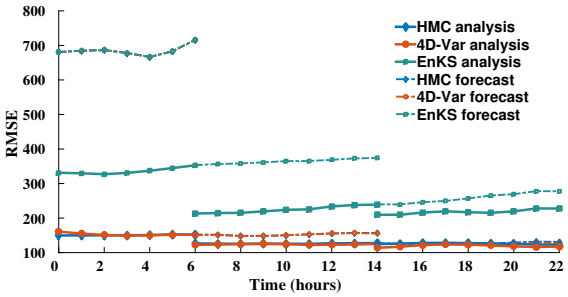
The symplectic integrator used with the HMC smoother is Verlet (3.12) with an empirically tuned step length of $T_* = 0.1$, with $h_* = 0.01$, and $m = 10$. A practically useful recipe is to perturb the step length of the symplectic integrator, a procedure that guarantees that the results obtained are not contingent on that specific selection of step settings [63, 60]. The step length h is perturbed with uniform random noise: $h := (1 + r) \times h_*$, $r \sim \mathcal{U}(-0.2, 0.2)$. It is important to notice that the step h is perturbed only once at the beginning of the Hamiltonian trajectory and kept fixed for all the m steps. This actually means that the length of the Hamiltonian trajectory T is perturbed for each proposal state while keeping the number of steps m constant such that at each run the step size h scales accordingly

The number of burn-in steps is set to 30. We noticed that the HMC smoother converges to the posterior in much fewer steps ($5-10$). Four generated states are discarded between each selected state in the ensemble to guarantee independence of the generated ensemble members.

Linear observation operator results Here we show the results of EnKS, 4D-Var, and the HMC sampling smoother in the presence of the linear observation operator (3.23a). The ensemble size for both EnKS and HMC sampling smoother here is set to 100 on each of the three assimilation windows. The RMS errors for EnKS, 4D-Var, and the HMC smoother over the three assimilation windows are shown in Figure 3.4. Figure 3.4(a) reports the case where the background error covariance matrix $\mathbf{B}_0$ is kept fixed, and Figure 3.4(b) shows the case where $\mathbf{B}_0$ is updated, at the beginning of each assimilation window, according to equation (3.24).



(a) Experiment with constant $\mathbf{B}_0$



(b) Experiment with hybrid $\mathbf{B}_0$ using (3.24)

Figure 3.4: Data assimilation results for two scenarios using linear observations (3.23a) are shown. The first panel 3.4(a) shows RMSE with B_0 being fixed. The second panel 3.4(b) shows RMSE with B_0 being updated using (3.24). The symplectic integrator used in both cases is Verlet (3.12) with step $T = 0.1$, where $h = 0.01$, and $m = 10$. The number of dropped states between selected samples is 4.

As demonstrated by the results in Figure 3.4, the positive impact of updating the background error covariances on the quality of the analysis, using (3.24), is obvious for EnKS. The quality of the analyses in both cases of 4D-Var, and HMC smoother, is very similar, however in the second case a slight reduction in RMSE is noticed along the entire trajectory. This is appreciated by Figure 3.5 zooming onto the RMSE results over the second assimilation window.

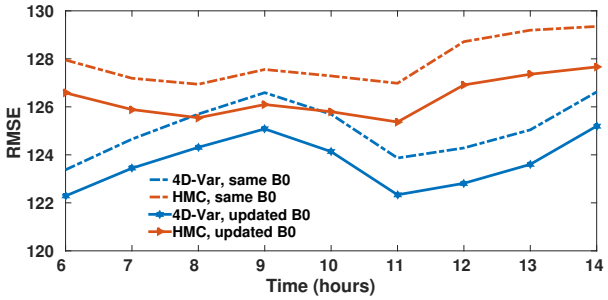
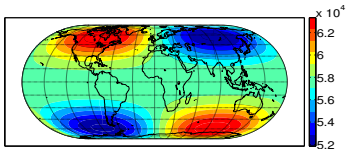


Figure 3.5: Same as results in Figure 3.4 with only RMS errors over the second window displayed for two scenarios. RMS errors obtained while keeping $\mathbf{B}_0$ fixed are plotted as dotted lines. RMS errors obtained with $\mathbf{B}_0$ being updated at the beginning of the assimilation window are plotted as dotted lines.

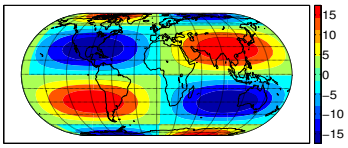
The HMC smoother can sample efficiently from the posterior distribution and the resulting analysis competes in accuracy with that obtained using 4D-Var. Figure 3.6 shows the three components at the beginning of the first window for the reference solution, the background state, 4D-Var analysis, HMC smoother analysis (ensemble mean), and EnKS analysis. Results shown in the analysis recovered from the noisy background by both 4D-Var and HMC smoother are almost identical. The results obtained using EnKS seem to improve over the consecutive cycles, but they don't compete with either 4D-Var or HMC smoother.

The assimilation results obtained over the next two windows with $\mathbf{B}_0$ kept fixed are shown in Figures 3.7 and 3.8. The performance of the two schemes, 4D-Var, and the HMC smoother is quite similar, and the HMC smoother analysis competes with the 4D-Var analysis, and both clearly outperform the EnKS.

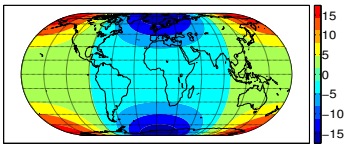
Updating the background error covariances can, in principle, enhance the performance of both the 4D-Var and the HMC smoother. In the case of 4D-Var, updating $\mathbf{B}_0$ results in lower RMSE which indicates that in real applications, the analysis is expected to be closer to reality. In addition to more accurate prior kernel, updating $\mathbf{B}_0$ will result in a better update of the mass matrix $\mathbf{M}$ which in turn is expected to result in better performance of the smoother. In our experiments the update has a small positive impact on the performance of the two data assimilation schemes as explained in Figures 3.4, 3.5. The positive effect here is explained by reduction in the RMS errors. The resulting ensemble-based forecast error covariance matrix that is used to update $\mathbf{B}_0$ can be crucial for cases where observations are sparse or not uniformly distributed over the grid, and therefore well worth the computational overhead of the forward propagation of all the analysis ensemble members to build



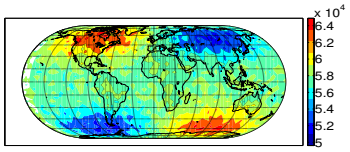
(a) Reference solution at the initial time, H component



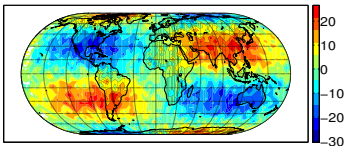
(b) Reference solution at the initial time, U component



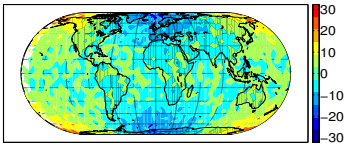
(c) Reference solution at the initial time, V component



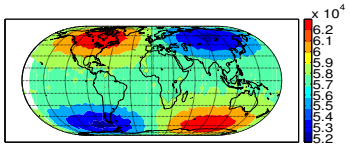
(d) Background solution at the initial time, H component



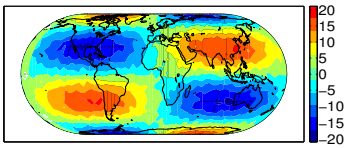
(e) Background solution at the initial time, U component



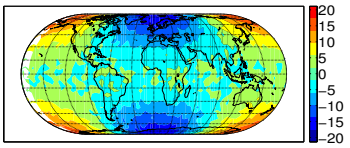
(f) Background solution at the initial time, V component



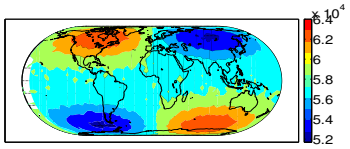
(g) HMC smoother analysis at the initial time, H component



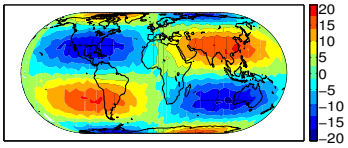
(h) HMC smoother analysis at the initial time, U component



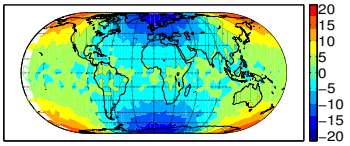
(i) HMC smoother analysis at the initial time, V component



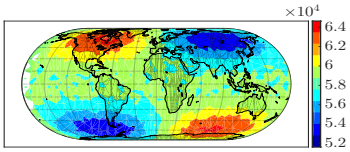
(j) 4D-Var analysis at the initial time, H component



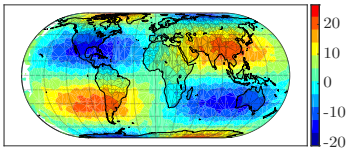
(k) 4D-Var analysis at the initial time, U component



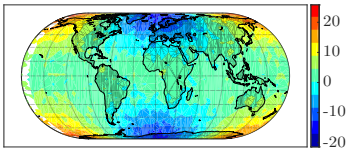
(l) 4D-Var analysis at the initial time, V component



(m) EnKS analysis at the initial time, H component

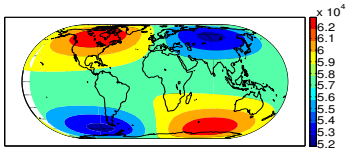


(n) EnKS analysis at the initial time, U component

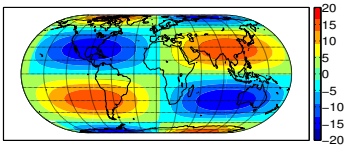


(o) EnKS analysis at the initial time, V component

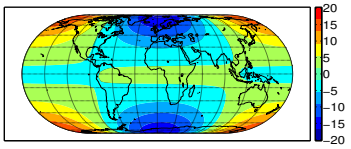
Figure 3.6: Four dimensional data assimilation results with linear observations (3.23a). The initial condition solutions at the beginning of the first window are shown. The data assimilation scheme and the state components are indicated under each panel. The assimilation window length is 6 hours, with hourly observations. The background error covariance matrix $\mathbf{B}_0$ is kept fixed.



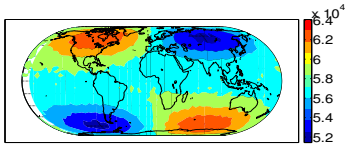
(a) Reference solution at the initial time, H component



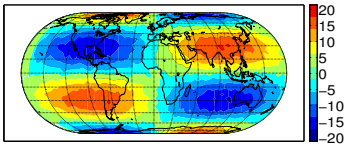
(b) Reference solution at the initial time, U component



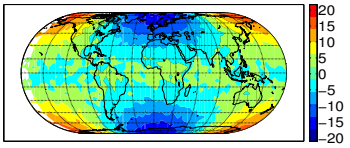
(c) Reference solution at the initial time, V component



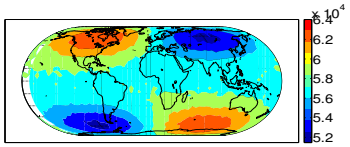
(d) HMC smoother analysis at the initial time, H component



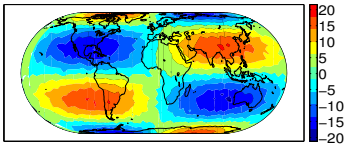
(e) HMC smoother analysis at the initial time, U component



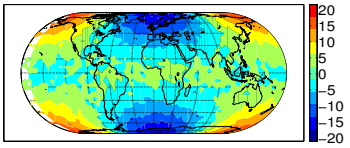
(f) HMC smoother analysis at the initial time, V component



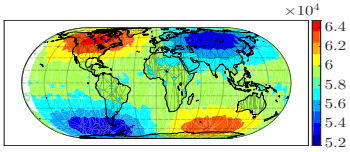
(g) 4D-Var analysis at the initial time, H component



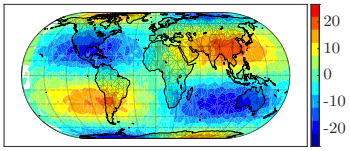
(h) 4D-Var analysis at the initial time, U component



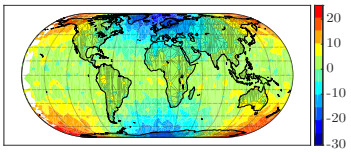
(i) 4D-Var analysis at the initial time, V component



(j) EnKS analysis at the initial time, H component

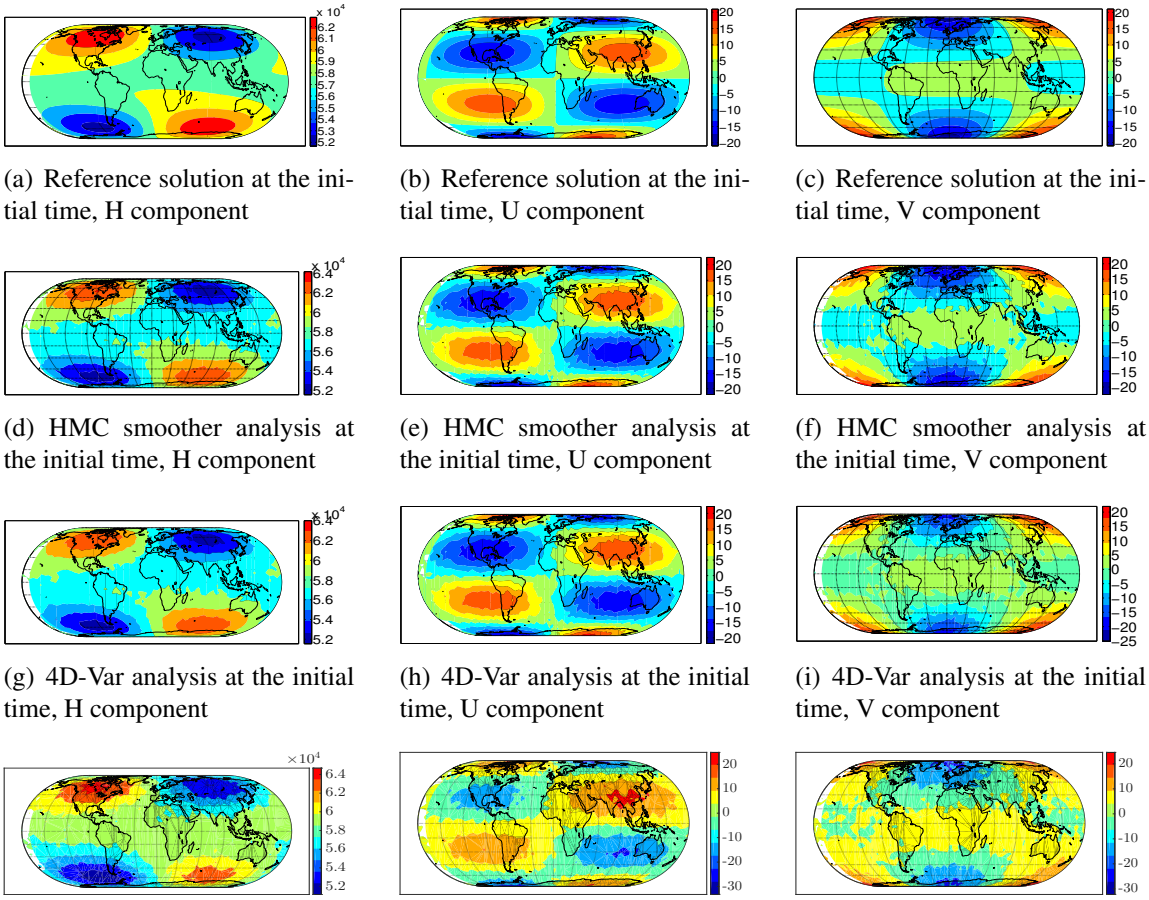


(k) EnKS analysis at the initial time, U component



(l) EnKS analysis at the initial time, V component

Figure 3.7: Four dimensional data assimilation results with linear observations (3.23a). The initial condition solutions at the beginning of the second window are shown. The data assimilation scheme and the state components are indicated under each panel. The assimilation window length is 8 hours, with hourly observations. The background error covariance matrix $\mathbf{B}_0$ is not updated.



(a) Reference solution at the initial time, H component

(b) Reference solution at the initial time, U component

(c) Reference solution at the initial time, V component

(d) HMC smoother analysis at the initial time, H component

(e) HMC smoother analysis at the initial time, U component

(f) HMC smoother analysis at the initial time, V component

(g) 4D-Var analysis at the initial time, H component

(h) 4D-Var analysis at the initial time, U component

(i) 4D-Var analysis at the initial time, V component

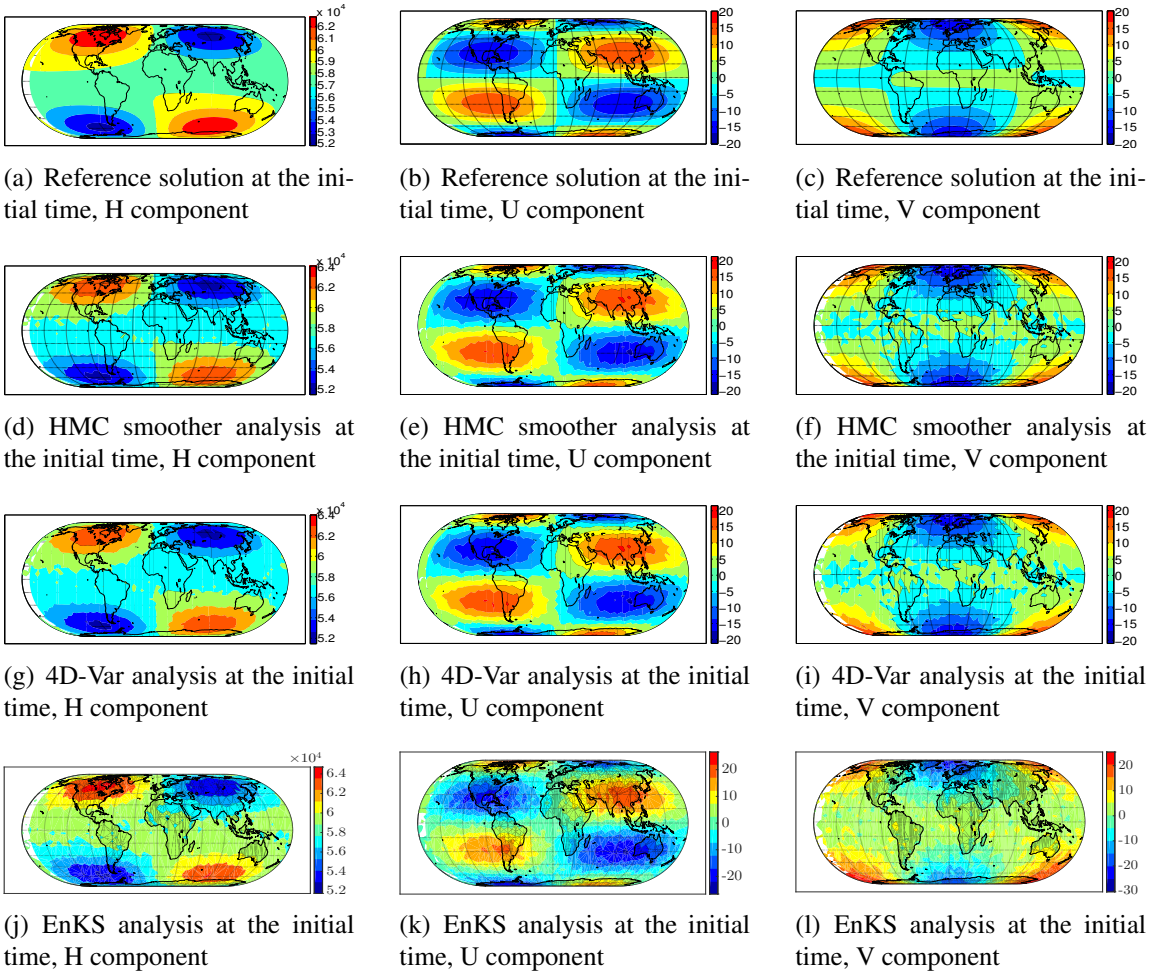
(j) EnKS analysis at the initial time, H component

(k) EnKS analysis at the initial time, U component

(l) EnKS analysis at the initial time, V component

Figure 3.8: Four dimensional data assimilation results with linear observations (3.23a). The initial condition solutions at the beginning of the third window are shown. The data assimilation scheme and the state components are indicated under each panel. The assimilation window length is 8 hours, with hourly observations. The background error covariance matrix $\mathbf{B}_0$ is kept fixed.

the full forecast ensemble. Results of the data assimilation process with hybrid (updated) background error covariance matrix on the next two windows are shown in Figures 3.9 and 3.10.



(a) Reference solution at the initial time, H component

(b) Reference solution at the initial time, U component

(c) Reference solution at the initial time, V component

(d) HMC smoother analysis at the initial time, H component

(e) HMC smoother analysis at the initial time, U component

(f) HMC smoother analysis at the initial time, V component

(g) 4D-Var analysis at the initial time, H component

(h) 4D-Var analysis at the initial time, U component

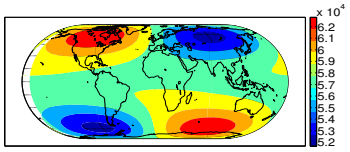
(i) 4D-Var analysis at the initial time, V component

(j) EnKS analysis at the initial time, H component

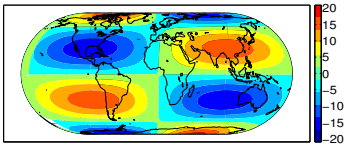
(k) EnKS analysis at the initial time, U component

(l) EnKS analysis at the initial time, V component

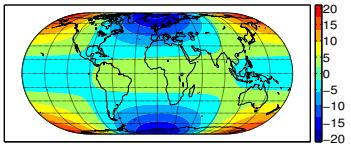
Figure 3.9: Four dimensional data assimilation results with linear observations (3.23a). The initial condition solutions at the beginning of the second window are shown. The data assimilation scheme and the state components are indicated under each panel. The assimilation window length is 8 hours, with hourly observations. The background error covariance matrix $\mathbf{B}_0$ is updated using (3.24) for both schemes.



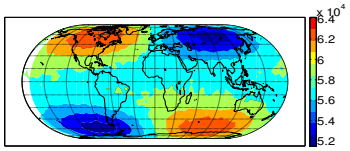
(a) Reference solution at the initial time, H component



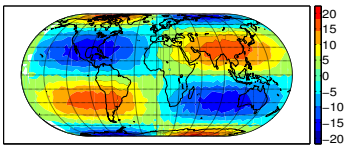
(b) Reference solution at the initial time, U component



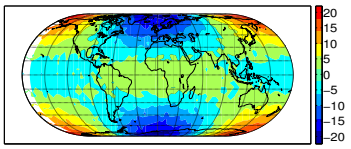
(c) Reference solution at the initial time, V component



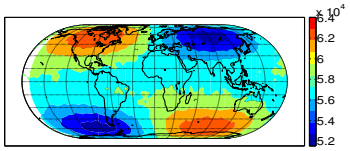
(d) HMC smoother analysis at the initial time, H component



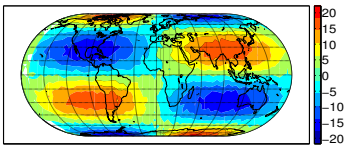
(e) HMC smoother analysis at the initial time, U component



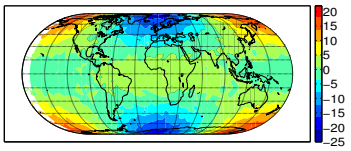
(f) HMC smoother analysis at the initial time, V component



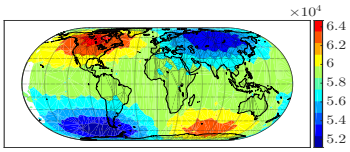
(g) 4D-Var analysis at the initial time, H component



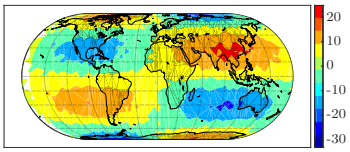
(h) 4D-Var analysis at the initial time, U component



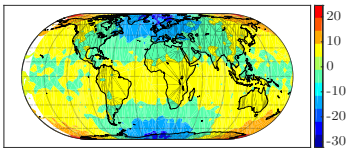
(i) 4D-Var analysis at the initial time, V component



(j) EnKS analysis at the initial time, H component



(k) EnKS analysis at the initial time, U component



(l) EnKS analysis at the initial time, V component

Figure 3.10: Four dimensional data assimilation results with linear observations (3.23a). The initial condition solutions at the beginning of the third window are shown. The data assimilation scheme and the state components are indicated under each panel. The assimilation window length is 8 hours, with hourly observations. The background error covariance matrix $\mathbf{B}_0$ is updated using (3.24) for both schemes.

Nonlinear observation operator results The HMC sampling smoother, EnKS, and 4D-Var were tested in the presence of a nonlinear observation operator (3.23b). The RMS errors obtained by EnKS, 4D-Var, and the HMC sampling smoother are shown in Figure 3.11. We tested the HMC smoother and EnKS using different ensemble sizes. Here we show the results using only 30 ensemble members. The background error covariance matrix $\mathbf{B}_0$ is updated at the beginning of each of the three assimilation cycles. The ensemble obtained by the HMC smoother is used to generate a flow-dependent background error covariance matrix $\mathbf{B}_0^{\text{ensemble}}$ that is substituted in (3.24) to obtain a hybrid version of $\mathbf{B}_0$ for both 4D-Var and the HMC smoother. The update of $\mathbf{B}_0$ in the case of EnKS is carried out using the flow-dependent covariances obtained from the ensembles generated by EnKS. As revealed by Figure 3.11, the analysis states generated by both 4D-Var and the HMC smoother demonstrate similar behaviour in terms of the RMS errors, and both behave much better than EnKS. It is worth mentioning here that the EnKS results presented here should not be taken as means to draw general conclusions about the performance of EnKS.

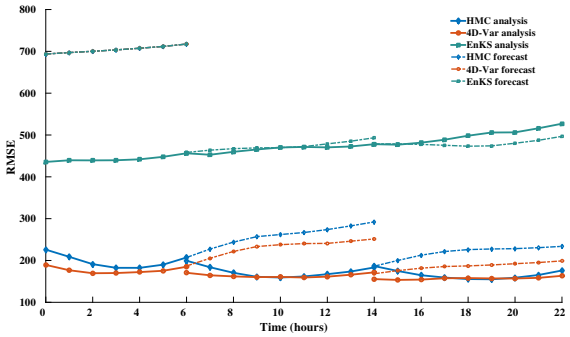
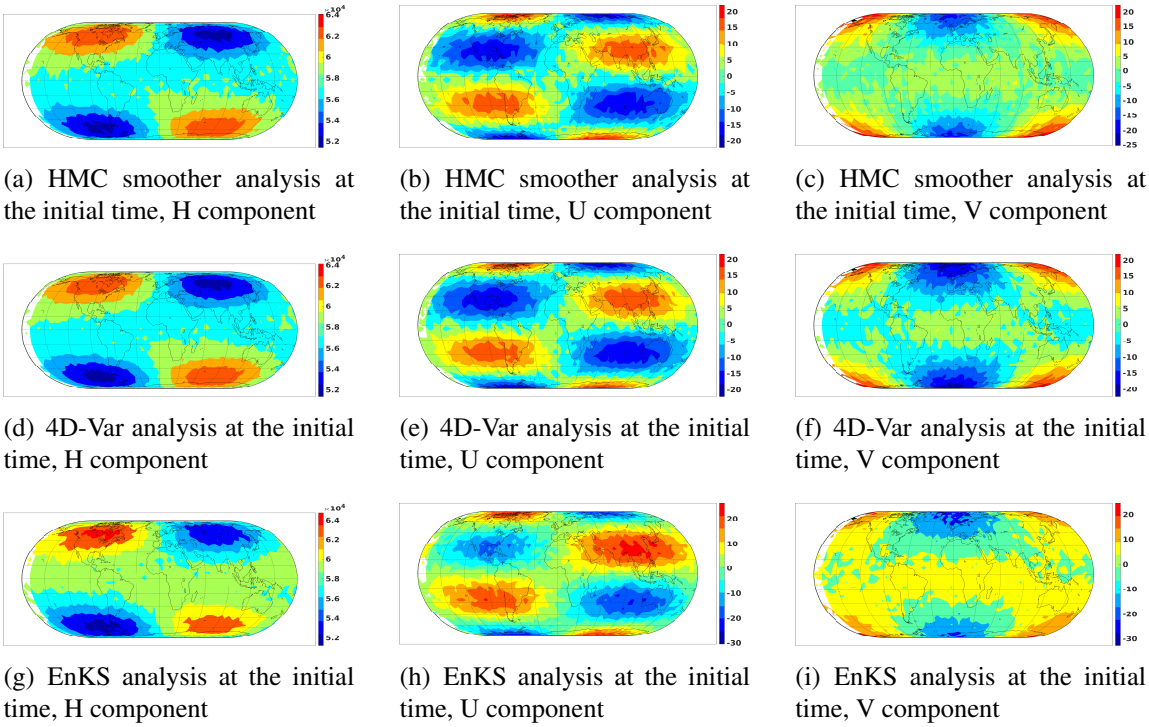


Figure 3.11: Data assimilation results using nonlinear observations (3.23b) are shown. The RMS errors are shown with B_0 being updated using (3.24). The ensemble size for HMC smoother and EnKS is set to 30. The symplectic integrator used is Verlet (3.12) with step $T = 0.1$, where $h = 0.01$, and $m = 10$. The number of dropped states between selected samples is 4.

Figure 3.12 shows the three components of the state vector obtained at the beginning of the third assimilation window for HMC smoother analysis (ensemble mean), 4D-Var analysis, and EnKS analysis. The results shown in Figure 3.12 meet our expectations based on the RMSE results demonstrated in Figure 3.11. While the initial conditions generated by 4D-Var and the HMC sampling smoother are similar, EnKS fails to recover the true solution given the noisy background and the noisy nonlinear observations. These results obtained using relatively small ensemble size in the presence of a highly nonlinear obser-

vation operator is promising and show good signs that the HMC smoother can accurately sample high dimensional posteriors. The information obtained using the HMC smoother however come at a cost. The current formulation of the HMC smoother is expensive. In subsection 3.4.3, we discuss the computational cost of the HMC smoother and present some ideas to reduce it's cost while maintaining the desired performance.



(a) HMC smoother analysis at the initial time, H component

(b) HMC smoother analysis at the initial time, U component

(c) HMC smoother analysis at the initial time, V component

(d) 4D-Var analysis at the initial time, H component

(e) 4D-Var analysis at the initial time, U component

(f) 4D-Var analysis at the initial time, V component

(g) EnKS analysis at the initial time, H component

(h) EnKS analysis at the initial time, U component

(i) EnKS analysis at the initial time, V component

Figure 3.12: Four dimensional data assimilation results with linear observations (3.23a). The initial condition solutions at the beginning of the third window are shown. The data assimilation scheme and the state components are indicated under each panel. The assimilation window length is 8 hours, with hourly observations. The background error covariance matrix $\mathbf{B}_0$ is updated using (3.24) for both schemes.

3.4.3 Computational considerations

Both 4D-Var and HMC smoother require the same computational infrastructure, namely, an adjoint model that computes the gradient of the 4D-Var cost functional (3.7b). This gradient calculation is the computational bottleneck for both 4D-Var and HMC smoother.

It requires forward propagation of the model, and a backward propagation of the adjoint model. Here we discuss the computation cost of the HMC smoother compared to 4D-Var in the presence of the linear observation operator (3.23a). In our shallow water test the cost of one adjoint model run is approximately equivalent to 2.5 times the cost of one forward model run. This makes the cost of gradient evaluation approximately equal to 3.5 the cost of a forward model run.

Table 3.1: Data assimilation scheme cost for the shallow water model with linear observations (3.23a). Number of function evaluations (forward model runs), and the number of optimization iterations (for 4D-Var) are listed. The cost of one adjoint run is approximately equal to 2.5 forward model runs in the current settings. Total cost is approximated in terms of number of forward model runs. The cost of the HMC sampling smoother is the same for the three assimilation windows.

Data assimilation scheme	**Cost**	**Assimilation window**				
		(1)	(2)		(3)	
		Fixed $\mathbf{B}_0$	Fixed $\mathbf{B}_0$	Hybrid $\mathbf{B}_0$	Fixed $\mathbf{B}_0$	Hybrid $\mathbf{B}_0$
4D-Var	Function evaluations	151	97	101	96	93
	Number of iterations	49	47	46	46	45
	Cost in equivalent forward model runs	322.5	261.5	262	257	250.5
HMC smoother	Number of proposed states	530				
	Cost per proposal	4.5				
	Cost in equivalent forward model runs	2,385				

The cost of the 4D-Var depends on the number of iterations and function evaluations required by the optimization algorithm. On the first window the number of iterations required by the LBFGS optimizer is 49, with 151 function evaluations. The total cost of the 4D-Var solution is then $151 + 49 \times 3.5 = 322.5$ equivalent forward model runs.

The cost of the HMC smoother depends on the configuration of the chain: the number of burn-in step, the number of dropped states at stationarity, and step-size settings of the symplectic integrator. The symplectic integrator itself controls the number of adjoint runs to evaluate the gradient of the cost functional in order to propose a new state to the chain. The size of the desired ensemble controls the length of the Markov chain and consequently the total cost of the analysis step by the HMC smoother. The Verlet integrator (3.12), used in the current experiments, requires a single adjoint run to propose a new state to the chain. The acceptance/rejection criterion requires an additional forward run to evaluate the loss of energy. This makes the cost of generating a proposal state to the Markov chain approximately equal to 4.5 the cost of a model run. On all assimilation windows

the chosen ensemble size is 100. The number of burn-in states is 30, and 4 states are rejected between consecutive selected samples. The HMC sampling smoother, in this case generates $30 + 100 \times 5 = 530$ states to collect the analysis ensemble, with a total cost roughly equal to $530 \times 4.5 = 2,385$ forward model runs.

The computational cost of the two data assimilation schemes, 4D-Var and HMC smoother in the presence of a linear observation operator (3.23a) , on each assimilation window are summarized in Table 3.1. Notice that the total cost of DA schemes is given in terms of the total number of forward model runs. On the first window the cost of the HMC smoother is approximately 9 times the cost of the 4D-Var scheme. On the next two windows, the HMC smoother costs roughly 11 times the cost of the 4D-Var. A cost-reduction in 4D-Var is expected because the forecast state is closer to the MAP than the case on the first window. The higher computational cost of the HMC smoother can be handled more efficiently by parallelizing the sampling scheme and the gradient calculations. Another way to reduce the computational cost of the HMC smoother is to replace the burn-in steps with a sub-optimal 4D-Var obtained using a small number of iterations. The computational cost of the proposed sampling smoother can of course by be reduced by decreasing the ensemble size, however this will result in higher sampling error. The impact of the sampling errors can be assessed by the techniques developed in [99, 100].

The increased cost of the HMC smoother could be acceptable in view of the additional useful information it provides: a sample estimate of the analysis probability distribution (and as immediate consequences an analysis error covariance matrix and a flow-dependent background error covariance matrix for the next cycle). Moreover, the computational efficiency of the HMC sampling smoother can be improved by following several strategies. One idea to reduce the cost of the HMC sampling smoother is to replace model dynamics with reduced-order counterpart. This idea is explored in detail in [39].

3.5 Conclusion and Future Work

A four-dimensional data assimilation smoother is proposed in this chapter. The smoother samples from the posterior distribution using a Hybrid Monte-Carlo approach. The 4D-Var approach provides a MAP estimate of the true state, but it does not compute a measure of uncertainty of the analysis. The HMC smoother builds an ensemble approximating the posterior PDF. This can be used to estimate the true state together with the uncertainty in analysis, e.g., by calculating the ensemble mean and ensemble-based analysis error covariance matrix. Moreover, propagating the analysis ensemble to the beginning of the next

assimilation window provides a forecast ensemble that can be used to construct a flow-dependent background covariance matrix for this new window. Unlike several popular hybrid approaches, the HMC smoother generates an analysis error covariance that is consistent with the analysis state – because both statistics are produced by one consistent data assimilation scheme.

The HMC smoother requires an adjoint of the simulation model, and runs on the same computational infrastructure as 4D-Var. The computational cost of the HMC smoother is - as of now - larger than that of 4D-Var. The efficiency issue must be addressed before the HMC smoother becomes fully practical. We are currently investigating several strategies to enhance the performance of the sampling smoother and to reduce its computational cost. Parallelizing the sampling smoother will be considered. We will also test the HMC smoother on the case of imperfect models, and non-Gaussian errors.

Chapter 4

The Reduced-Order Hybrid Monte Carlo Sampling Smoother

4.1 Introduction

Many large-scale prediction problems such as atmospheric forecasting are formulated as initial value problems. The uncertainty of the associated model initial conditions can be decreased by combining imperfect forecasts produced by propagating the model dynamics with real measurements collected at discrete times over an assimilation window. The model state and the observations are both uncertain and can be modeled as random variables. The probability distribution describing the knowledge about the initial state system, before incorporating observations information, is known as the prior distribution. The likelihood function accounts for the discrepancies between the measurements and model-predicted observations. Data assimilation applies Bayes' theorem to obtain a posterior probability distribution, named the analysis, that characterizes the knowledge about the system state given the observed measurements. In practice, due to the state space high-dimensionality of many realistic models, it is impossible to exactly describe the posterior distribution and several assumptions and approximations are necessary. Widely accepted assumptions are that the background and the observation errors are characterized by Gaussian distributions, with no correlations between observation errors at different time instances.

Two families of methodologies are generally followed in order to generate accurate estimates of the true system state. Ensemble-based statistical methods seek to approximate the

posterior probability density function (PDF) based on an ensemble of model states, while the variational approaches estimate the true state of the system by searching for the state that maximizes the posterior PDF. Among the variational techniques, the 4D-Var method achieves this goal by searching for a local minimum of an objective function corresponding to the negative logarithm of the posterior distribution. 4D-Var finds the maximum posterior (MAP) estimate of the true state and does not directly estimate the uncertainty associated with the analysis state. For scenarios including nonlinear observation operators and state models, the analysis distribution is not Gaussian, and the 4D-Var algorithm may be trapped in a local minimum of the cost function leading to incorrect conclusions.

Accurate solution of the non-Gaussian data assimilation problems requires accounting for all regions of high probability in the posterior distribution. The recently developed hybrid Monte-Carlo (HMC) sampling smoother [38, 101] is a four dimensional data assimilation scheme designed to solve the non-Gaussian smoothing problem by sampling from the posterior distribution. It relies on an accelerated Markov chain Monte-Carlo (MCMC) methodology where a Hamiltonian system is used to formulate proposal densities. Two issues are important when using HMC sampling strategy. First, it requires the formulation of the posterior negative-log function gradient. Secondly, the involved Hamiltonian system is propagated using a symplectic integrator whose parameters must be carefully tuned in order to achieve good performance.

While producing consistent description of the updated system uncertainty (e.g., the analysis error covariance matrix), the original formulation of the HMC sampling smoother is computationally expensive when compared to the 4D-Var approach. This is due to the large number of gradient evaluations required, which translates into many forward and adjoint models runs. In the case of large scale problems the computational cost becomes prohibitive. In this present study we propose a practical solution by approximating the gradient using information obtained from lower-dimensional subspaces via model reduction.

Reduced order modeling refers to the development of low-dimensional systems that represent the important characteristics of a high-dimensional or infinite dimensional dynamical system. Typically this is achieved by projecting model dynamics onto a lower dimensional spaces. Construction of low relevant manifolds can be achieved using the reduced basis method [102, 103, 104, 105, 106, 107], dynamic mode decomposition [108, 109, 110, 111] and Proper Orthogonal Decomposition (POD) [112, 113, 114, 115]. The latest is the most prevalent basis selection method for nonlinear problems. Data analysis using POD and method of snapshots [116, 117, 118] is conducted to extract basis functions, from experimental data or detailed simulations of high-dimensional systems, for subsequent use in Galerkin projections that yield low dimensional dynamical models. By coupling POD with

empirical interpolation method (EIM) [102], discrete variant DEIM [119, 120, 121, 122] or best points interpolation method [123], one can obtain fast approximations of reduced order nonlinear terms independent of the dimension of high-fidelity space. Other such approaches include missing point estimation [124] and Gauss-Newton with approximated tensors [125, 126] relying upon the gappy POD technique [127].

The present manuscript develops practical versions of the HMC sampling smoother using approximate dynamical information obtained via POD/DEIM reduced order models [128, 129, 130]. Two reduced order variants are proposed differentiated by the choice of the sampling space used to generate the proposals. In the first case we are sampling in a reduced order space while the second approach samples directly from the high fidelity space. For both scenarios the negative logarithm of the posterior distribution reassembles one of the flavors of reduced order 4D-Var objective functions [131, 132, 133], where the prior term is either estimated in the reduced or full space respectively. While reduced order models are employed to estimate the posterior distribution negative logarithm and it's gradient, the associated Hamiltonian system generates proposals in a reduced or high-fidelity space depending on the nature of the sampling space.

The choice of reduced order manifolds is crucial for the accuracy of the reduced samplers. The smoothers may suffer from the fact that the basis elements are computed from a reference trajectory containing features which are quite different from those of the current proposal trajectory. To overcome the problem of unmodelled dynamics in the POD-basis we propose to update the basis from time to time according to the current state similarly as in the case of adaptive reduced order optimization [134, 135, 136]. Inspired by the recent advances in reduced order data assimilation field where it was shown that accurate reduced order Karush-Kuhn-Tucker conditions with respect to their full order counterparts highly increase the accuracy of the obtained analysis [137], we chose to update the reduced order manifolds based on high-fidelity forward and adjoint trajectories corresponding to current proposal as well as the associated gradient of the negative logarithm of the high-fidelity posterior distribution. The basis is refreshed once after several HMC smoothers iterations. The numerical results using swallow-water equations model on Cartesian coordinates reveal that the reduced-order versions of the smoother are accurately capturing the posterior probability density, while being significantly faster than the original full order formulation.

The chapter is organized as follows. Section 4.2 reviews the four-dimensional data assimilation problem and the original HMC smoother. Section 4.3 reviews reduced-order modeling, and introduces the reduced order 4D-Var data assimilation framework. Section 4.4 presents a rigorous description of the proposed versions of the HMC smoother. In section 4.5 several theoretical properties of the distributions sampled with reduced order models are derived. Numerical results are presented in Section 4.6 while conclusions are

drawn in Section 4.7.

4.2 Data Assimilation

One can solve a data assimilation (DA) problem by describing the posterior distribution in the Bayesian formalism given the prior and the likelihood function. In the four-dimensional DA (4DDA) context, the main goal is to describe the posterior distribution of system state at the initial time of a specific assimilation window. The assimilation window is defined based on the availability of observations at specific discrete time instances. The knowledge about the system at the initial time t_0 defines the distribution (the prior) $\mathcal{P}^{\mathrm{b}}(\mathbf{x}_0)$ of the state $\mathbf{x}_0 \in \mathbb{R}^{\mathrm{N_{state}}}$ before absorbing knowledge presented by the observations captured over the assimilation window, where $\mathrm{N_{state}}$ is the dimension of the model state space. Based on a set of observations $\{\mathbf{y}_k = \mathbf{y}[t_k] \in \mathbb{R}^{\mathrm{N_{obs}}}\}_{k=0,1,\dots,m}$, at the discrete time points $\{t_k\}_{k=0,1,\dots,m}$ in the interval $[t_0\,, t_F]$, the sampling distribution (likelihood function) is defined as $\mathcal{P}(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_m|\mathbf{x}_0)$, and the posterior distribution describing the updated information about the model state at the initial time, given the measurements, takes the general form

$$\mathcal{P}(\mathbf{x}_0|\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_m) = \frac{\mathcal{P}^{\mathrm{b}}(\mathbf{x}_0)\,\mathcal{P}(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_m|\mathbf{x}_0))}{\mathcal{P}(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_m)}\,. \tag{4.1}$$

Here $\mathrm{N_{obs}}$ is the dimension of the observation space, and $m+1$ is the number of observation time instances within an assimilation window.

Fully and accurately describing this general posterior probability density function in DA literature is an intractable problem, and usually simplifying assumptions are generally made. As we mentioned in Section 4.1, a frequent supposition is that the background and observation errors are characterized by Gaussian distributions. If the observations are assumed to be independent from the model states, and the associated error characteristics of these observations are not correlated in time, the posterior distribution takes the form

$$\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0) = \mathcal{P}(\mathbf{x}_0|\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_m) \propto \exp\left(-\mathcal{J}\mathbf{x}_0)\right), \tag{4.2a}$$

$$\mathbf{J} : \mathbb{R}^{\mathrm{N_{state}}} \to \mathbb{R}, \quad \mathcal{J}\mathbf{x}_0) = \frac{1}{2}||\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}}||^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m} ||\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)||^2_{\mathbf{R}_k^{-1}}\,, \tag{4.2b}$$

where $\mathbf{x}_0^{\mathrm{b}} = \mathbf{x}^{\mathrm{b}}[t_0]$ is a background/forecast state, the matrices $\mathbf{B}_0$ and $\mathbf{R}_k$, $k = 0, 1, \dots, m$, are the covariance matrices associated with the background and measurement errors respectively. The observation operator $\mathcal{H}_k : \mathbb{R}^{\mathrm{N_{state}}} \to \mathbb{R}^{\mathrm{N_{obs}}}$, at time instance t_k, maps a given

state $\mathbf{x}_k \in \mathbb{R}^{\mathrm{N_{state}}}$ to the observation space. The dimension of the observation space is usually much smaller than the size of the state space, that is $\mathrm{N_{obs}} \ll \mathrm{N_{state}}$. The associated norms over the state and observation spaces are defined as:

$$\|\mathbf{a} - \mathbf{b}\|_{\mathbf{C}}^2 = (\mathbf{a} - \mathbf{b})^T \mathbf{C} (\mathbf{a} - \mathbf{b}), \tag{4.3}$$

where $\mathbf{a}$, $\mathbf{b}$ belong to either $\mathbb{R}^{\mathrm{N_{state}}}$ or $\mathbb{R}^{\mathrm{N_{obs}}}$ and $\mathbf{C}$ is a matrix of $\mathbb{R}^{\mathrm{N_{state}} \times \mathrm{N_{state}}}$ or $\mathbb{R}^{\mathrm{N_{obs}} \times \mathrm{N_{obs}}}$ dimensions.

The model state $\mathbf{x}_k = \mathbf{x}[t_k]$, is obtained from the initial state $\mathbf{x}_0$, by propagating the model dynamics to time point t_k, i.e.

$$\mathbf{x}_k = \mathcal{M}_{t_0 \to t_k}(\mathbf{x}_0) = \mathcal{M}_{0,k}(\mathbf{x}_0)\,, \tag{4.4}$$

where, $\mathcal{M}_{t_0 \to t_k} : \mathbb{R}^{\mathrm{N_{state}}} \to \mathbb{R}^{\mathrm{N_{state}}}$, $k = 1, \ldots, m$ represents the discretized mathematical model reflecting the state dynamics. The function $\mathbf{J}$ given in (4.2) is quadratic, and consequently the distribution (4.2) is Gaussian distribution, only if the states $\mathbf{x}_k$ are linearly related to the system initial condition $\mathbf{x}_0$, and the observations $\mathbf{y}_k$ are linearly related to the model states $\mathbf{x}_k$. In virtually all practical settings, the time dependent models are complicated resulting in a nonlinear bond between states at different time instances. More difficulty is added since for example, in the field of atmospheric sciences, highly-nonlinear observation operators are often constructed to relate new types of measurements to state variables.

4.2.1 4D-Var data assimilation

The strongly-constrained 4D-Var DA scheme is an optimization algorithm that searches for the maximum a posteriori estimate (MAP) by seeking a *local* minimizer of the cost function (4.2b), constrained by the model equation (4.4). Precisely, the constrained optimization problem is defined as

$$\begin{aligned} \min_{\mathbf{x}_0} \mathcal{J}(\mathbf{x}_0) &= \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{b}}\|_{\mathbf{B}_0^{-1}}^2 + \frac{1}{2}\sum_{k=0}^{m} \|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)\|_{\mathbf{R}_k^{-1}}^2\,, \\ \mathbf{x}_k &= \mathcal{M}_{t_0 \to t_k}(\mathbf{x}_0)\ \ k = 1, \ldots, m. \end{aligned} \tag{4.5}$$

Gradient-based schemes are generally employed to find the corresponding initial conditions which in the control theory language are known as control variables. Using the Lagrange multiplier technique the constrained optimization problem (4.5) is redesigned as an

unconstrained minimization problem with the associated first order necessary optimality conditions:

$$\begin{aligned}
&\textit{Forward model}: && \mathbf{x}_k = \mathcal{M}_{t_0\to t_k}(\mathbf{x}_0),\ k = 1,\dots,m, && \text{(4.6a)}\\
&\textit{Adjoint model}: && \boldsymbol{\lambda}_m = \mathbf{H}_m^T\mathbf{R}_m^{-1}(\mathbf{y}_m - \mathcal{H}_m(\mathbf{x}_m)), && \text{(4.6b)}\\
& && \boldsymbol{\lambda}_k = \mathbf{M}_{t_0\to t_{k+1}}^T\boldsymbol{\lambda}_{k+1} + \mathbf{H}_k^T\mathbf{R}_k^{-1}(\mathbf{y}_k - \mathcal{H}\mathbf{x}_k)), && \text{(4.6c)}\\
&\textit{Cost function gradient:} && \nabla_{\mathbf{x}_0}\mathcal{J}(\mathbf{x}_0) = -\mathbf{B}_0^{-1}(\mathbf{x}_0^{\mathrm{b}} - \mathbf{x}_0) - \boldsymbol{\lambda}_0 = 0\,, && \text{(4.6d)}\\
& && k = m-1,..,0\,. &&
\end{aligned}$$

The Jacobians of the model and observation operators are denoted by $\mathbf{M}_{t_0\to t_{k+1}}$, $k = 0, 1, \dots, m-1$ and $\mathbf{H}_k$, $k = 0, 1, \dots, m$ while the adjoint solution $\boldsymbol{\lambda}_k \in \mathbb{R}^{\mathrm{N_{state}}}$, $k = 0, 1, \dots, m$, provides an efficient way to compute the gradient (4.6d). The nonlinear optimization procedure is computationally expensive and either low-resolution models (incremental 4D-Var [138]), or alternatively reduced-order models are used [137] to alleviate this drawback. It is well-known that 4D-Var does not inherently provide a measure of the uncertainty about the updated state (e.g., analysis error covariance matrix) and usually hybrid methods are considered to account for this type of information. This approach results in some inconsistency between the analysis state and the analysis error covariance matrix especially when they are obtained using different algorithms.

4.2.2 Smoothing by sampling and the HMC sampling smoother

Monte-Carlo smoothing refers to the process of representing/approximating the posterior distribution (4.2) using an ensemble of model states sampled from that posterior. Ensemble Kalman smoother (EnKS) [91] is an extension of the well-known ensemble Kalman Filter [53] to the case where observation are assimilated simultaneously. EnKS produces a minimum-variance unbiased estimate (MVUE) of the system state by estimating the expectation of the posterior $\mathrm{E}_{\mathcal{P}^{\mathrm{a}}}[\mathbf{x}_0]$ using the mean of an ensemble of states. The strict Gaussianity and linearity assumptions imposed by EnKS, usually result in poor performance of the smoother.

Pure sampling of the posterior distribution (4.1) using a Markov Chain Monte-Carlo (MCMC) [58, 60] technique is known in theory to provide more accurate estimates without strictly imposing linearity or Gaussianity constraints. MCMC is a family of Monte-Carlo schemes tailored to sample a given distribution (up to a proportionality constant), by constructing a Markov chain whose stationary distribution is set as the target distribution. By design, an MCMC sampler is guaranteed to converge to its stationarity. However, the choice of

the proposal density, the convergence rate, the acceptance rate, and the correlation level among sampled points are the main building blocks of MCMC responsible for its performance and efficiency. Practical application of MCMC requires developing accelerated chains those can attain stationarity fast, and then explore the state space efficiently in very few steps. One of the MCMC samplers mainly designed for complicated PDFs and large dimensional spaces is the Hamiltonian/Hybrid Monte-Carlo sampler (HMC). HMC was firstly presented in [48] as an accelerated MCMC sampling algorithm. The sampler mainly uses information about the geometry of the posterior to guide its steps in order to avoid random walk behaviour and visit more frequently regions with high probability with the capability of jumping between separated modes of the target PDF.

HMC sampling: As all other MCMC samplers, HMC samples from a PDF

$$\pi(\mathbf{x}) \propto \exp\left(-\mathbf{J}^N(\mathbf{x})\right);\ \mathbf{x} \in \mathbf{R}^{\mathrm{N_{state}}}\,, \tag{4.7}$$

where $\exp\left(-\mathbf{J}^N(\mathbf{x})\right)$ is the shape function of the distribution, and $\mathbf{J}^N : \mathbb{R}^{\mathrm{N_{state}}} \to \mathbb{R}$ is the PDF negative-log. The power of MCMC, and consequently HMC, is that only the shape function, or alternatively the negative-log, is needed, while the scaling factor is not strictly required as in the case of standard application of Bayes' theorem. HMC works by viewing $\mathbf{x}$ as a position variable in an extended phase space consisting of points $(\mathbf{p}, \mathbf{x}) \in \mathbf{R}^{2\mathrm{N_{state}}}$, where $\mathbf{p} \in \mathbb{R}^{\mathrm{N_{state}}}$ is an auxiliary momentum variable. The Hamiltonian dynamics is modeled by the set of ordinary differential equations (ODEs):

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \nabla_{\mathbf{p}} H\,, \\ \frac{d\mathbf{p}}{dt} &= -\nabla_{\mathbf{x}} H\,, \end{aligned} \tag{4.8}$$

where $H = H(\mathbf{p}, \mathbf{x})$ is the constant total energy function of the system, known as the the Hamiltonian function or simply the Hamiltonian. A standard formulation of the Hamiltonian in the context of HMC is:

$$H(\mathbf{p}, \mathbf{x}) = \underbrace{\frac{1}{2}\,\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}}_{\text{kinetic energy}} + \underbrace{\mathcal{J}^N(\mathbf{x})}_{\text{potential energy}}\,, \tag{4.9}$$

where $\mathbf{M} \in \mathbb{R}^{\mathrm{N_{state}}\times\mathrm{N_{state}}}$ is a positive definite matrix known as the mass matrix. This particular formulation leads to a canonical distribution of the joint state ($\mathbf{p}$, $\mathbf{x}$) proportional to:

$$\exp\left(-H(\mathbf{p},\ \mathbf{x})\right) = \exp\left(-\frac{1}{2}\,\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}\right)\,\pi(\mathbf{x})\,. \tag{4.10}$$

The exact flow $\Phi_T : \mathbb{R}^{2\mathrm{N_{state}}} \to \mathbb{R}^{2\mathrm{N_{state}}}; \quad \Phi_T(\mathbf{p}[0], \mathbf{x}[0]) = (\mathbf{p}[T], \mathbf{x}[T])$ describes the time evolution of the Hamiltonian system (4.8) and is practically approximated using a numerical integrator that is symplectic as $\phi_T : \mathbb{R}^{2\mathrm{N_{state}}} \to \mathbb{R}^{2\mathrm{N_{state}}}; \quad \phi_T(\mathbf{p}[0], \mathbf{x}[0]) \approx (\mathbf{p}[T], \mathbf{x}[T])$. The use of a symplectic numerical integrator to approximate the exact Hamiltonian flow results in changes of total energy. Traditional wisdom recommends splitting the pseudo-time step T into m smaller steps of size h in order to simulate the Hamiltonian trajectory between points $(\mathbf{p}[0], \mathbf{x}[0])$, $(\mathbf{p}[T], \mathbf{x}[T])$ more accurately. The symplectic integrator of choice is position (or velocity) Verlet. New higher order symplectic integrators were proposed recently and tested in the context of filtering for data assimilation [62, 70]. One step of size h of the position Verlet [61, 62] integrator is described as follows

$$\mathbf{x}[h/2] = \mathbf{x}[0] + \frac{h}{2}\,\mathbf{M}^{-1}\,\mathbf{p}[0]\,, \tag{4.11a}$$

$$\mathbf{p}[h] = \mathbf{p}[0] - h\,\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}[h/2])\,, \tag{4.11b}$$

$$\mathbf{x}[h] = \mathbf{x}[h/2] + \frac{h}{2}\,\mathbf{M}^{-1}\,\mathbf{p}[h]. \tag{4.11c}$$

While, the mass matrix is a user-defined parameter, it can be designed to enhance the performance the sampler [70]. The step parameters of the symplectic integrator m, h can be empirically chosen by monitoring the acceptance rate in a preprocessing step. Specifically, the parameters of the Hamiltonian trajectory can be empirically adjusted such as to achieve a specific rejection rate. Generally speaking, the step size should be chosen to achieve a rejection rate between 25% and 30%, and the number of steps should generally be large [60].

Adaptive versions of HMC have been also proposed with the capability of adjusting it's step parameters. No-U-Turn sampler (NUTS) [74] is a version of HMC capable of automatically tuning its parameters to prohibit the sampler from retracing its steps along the constructed Hamiltonian trajectory. Another HMC sampler that tunes its parameters automatically using third-order derivative information is the Riemann manifold HMC (RMHMC) [65].

The intuition behind HMC sampler is to build a Markov chain whose stationary distribution is defined by the canonical PDF (4.10). In each step of the chain, a random momentum $\mathbf{p}$ is drawn from a Gaussian distribution $\mathcal{N}(0, \mathbf{M})$, and the Hamiltonian dynamics (4.8) at the final pseudo-time interval proposes a new point that is either accepted or rejected using a Metropolis-Hastings criterion. The two variables $\mathbf{p}$, and $\mathbf{x}$ are independent, so discarding the momentum generated at each step will leave us with sample points $\mathbf{x}$ generated from our target distribution.

In a previous work [101] we proposed using HMC as a pure sampling smoother to solve

the nonlinear 4DDA smoothing problem. The method samples from the posterior distribution of the model state at the initial time on an assimilation window on which a set of observations are given at discrete times. Following general assumptions where the prior is Gaussian and the observation errors are normally distributed, the target distribution defined in (4.7) is identical with the posterior distribution associated with the smoother problem (4.2). Consequently the PDF negative-log $\mathbf{J}^N$ in (4.7) resembles the 4D-Var cost function $\mathbf{J}$ defined in (4.5) and the gradient of the potential energy required by the symplectic integrator is the gradient of the 4D-Var cost functional (4.6d). The main hindrance stems from the requirement of HMC to evaluate the gradient of the potential energy (target PDF negative-log) at least as many times as the symplectic integrator is involved, which is an expensive process. Despite the associated computational overhead, the numerical results presented in [101] show the potential of using HMC smoother to sample multi-modal, high-dimensional posterior distributions formulated in the smoothing problem.

4.3 Four-Dimensional Variational Data Assimilation with Reduced-Order Models

Optimization problems such as the one described in (4.5) for nonlinear partial differential equations often demand very large computational resources, so that the need for developing fast novel approaches emerges. Recently the reduced order approach applied to optimal control problems for partial differential equations has received increasing attention. The main idea is to project the dynamical system onto subspaces consisting of basis elements that represent the characteristics of the expected solution. These low order models serve as surrogates for the dynamical system in the optimization process and the resulting approximate optimization problems can be solved efficiently.

4.3.1 Reduced order modeling

Reduced order modeling refers to the development of low-dimensional models that represent desired characteristics of a high-dimensional or infinite dimensional dynamical system. Typically, models are constructed by projection of the high-order, high-fidelity model onto a suitably chosen low-dimensional reduced-basis [139]. Most reduced-bases for nonlinear problems are constructed from a collection of simulations (methods of snapshots [116, 117, 118])

The most popular nonlinear model reduction technique is Proper Orthogonal Decompo-

sition (POD) and it usually involves a Galerkin projection with basis $V \in \mathbb{R}^{\mathrm{N_{state}} \times \mathrm{N_{RED}}}$ obtained as the output of Algorithm 5. Here NRED is the dimensional of the reduced-order state space spanned e.g., by the POD basis.

Algorithm 5 POD basis construction

1: Solve for the state variable solutions $\mathbf{x}_k,\ k = 1, .., m$ of (4.4). One can make use of more snapshots to construct the basis thus for example to consider a number of time steps larger than m.
2: Compute the singular value decomposition (SVD) for the state variable snapshots matrix $[\mathbf{x}_0\, \mathbf{x}_1\ \ \ldots\ \ \mathbf{x}_m] = \bar{\mathbf{V}}\Sigma\bar{W}^T$, with the singular vectors matrix $\bar{\mathbf{V}} = [\mathbf{v}_i]_{i=1,..,\mathrm{N_{state}}}$.
3: Using the singular-values $\lambda_1 \geq \lambda_2 \geq ... \lambda_n \geq 0$ stored in the diagonal matrix Σ, define $I(p) = (\sum_{i=1}^{p} \lambda_i)/(\sum_{i=1}^{\mathrm{N_{state}}} \lambda_i)$.
4: Choose NRED, the dimension of the POD basis, such that $\mathrm{N_{RED}} = \min_p\{I(p) : I(p) \geq \gamma\}$ where $0 \leq \gamma \leq 1$ is the percentage of total information captured by the reduced space $\mathcal{X}^{\mathrm{N_{RED}}} = \mathrm{range}(\mathbf{V})$, usually $\gamma = 0.99$.

Assuming a POD expansion $\mathbf{x}_k \approx \mathbf{V}\tilde{\mathbf{x}}_k,\ \tilde{\mathbf{x}}_k \in \mathbb{R}^{\mathrm{N_{RED}}},\ k = 0, .., m$ (for simplicity we neglected the centering trajectory, shift mode or mean field correction [140]) and making use of the basis orthogonality the associated POD-Galerkin model of (4.4) is obtained as

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{V}^T \mathcal{M}_{t_k \to t_{k+1}}(\mathbf{V}\tilde{\mathbf{x}}_k),\ k = 0, .., m. \tag{4.12}$$

The efficiency of POD - Galerkin technique is limited to the linear or bilinear terms [141] and strategies such as Empirical Interpolation Method (EIM) [142], Discrete Empirical Interpolation Method (DEIM) [121, 128] and tensorial POD [141]are usually employed to alleviate this deficiency.

4.3.2 Reduced order 4D-Var data assimilation

The "adjoint of reduced plus reduced of adjoint" approach (ARRA) leads to the construction of consistent feasible reduced first order optimality conditions [137] and this framework is employed to build the reduced POD manifolds for the reduced HMC samplers. In the case of Galerkin projection the POD reduced space is constructed based on sampling of both full forward and adjoint trajectories as well as the gradient of the cost function background term.

The reduced data assimilation problem minimizes the following reduced order cost func-

tion $\mathbf{J}^{\text{POD}} : \mathbb{R}^k \to \mathbb{R}$

$$\mathbf{J}^{\text{POD}}(\tilde{\mathbf{x}}_0) = \frac{1}{2}\|\mathbf{V}\tilde{\mathbf{x}}_0 - \mathbf{x}_0^{\text{b}}\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m}\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{V}\tilde{\mathbf{x}}_k)\|^2_{\mathbf{R}_k^{-1}}\,, \tag{4.13a}$$

subject to the constraints posed by the ROM projected nonlinear forward model dynamics (4.12)

$$\tilde{\mathbf{x}}_{k+1} = \widetilde{\mathcal{M}}_{t_k\to t_{k+1}}(\tilde{\mathbf{x}}_k),\ \widetilde{\mathcal{M}}_{t_k\to t_{k+1}}(\tilde{\mathbf{x}}_k) = \mathbf{V}^T\mathcal{M}_{t_k\to t_{k+1}}(\mathbf{V}\tilde{\mathbf{x}}_k),\ k = 0, 1, \ldots, m\,. \tag{4.13b}$$

An observation operator that maps directly from the reduced model space to observations space may be introduced, however for this study the operator requires the projected states as input.

The associated reduced Karush-Kuhn-Tucker conditions [137] are:

ARRA reduced forward model: (4.14a)

$$\tilde{\mathbf{x}}_{k+1} = \widetilde{\mathcal{M}}_{t_k\to t_{k+1}}(\tilde{\mathbf{x}}_k)\,,\quad k = 0, .., m\,,$$

ARRA reduced adjoint model: (4.14b)

$$\tilde{\boldsymbol{\lambda}}_m = \mathbf{V}^T\widehat{\mathbf{H}}_m^T\mathbf{R}_m^{-1}(\mathbf{y}_m - \mathcal{H}_m(\mathbf{V}\tilde{\mathbf{x}}_m)),$$
$$\tilde{\boldsymbol{\lambda}}_k = \mathbf{V}^T\widehat{\mathbf{M}}_{t_k\to t_{k+1}}^T\mathbf{V}\tilde{\boldsymbol{\lambda}}_{k+1} + \mathbf{V}^T\widehat{\mathbf{H}}_k^T\mathbf{R}_k^{-1}(\mathbf{y}_k - \mathcal{H}_k(\mathbf{V}\tilde{\mathbf{x}}_k)),\quad k = m-1, .., 0,$$

ARRA cost function gradient : (4.14c)

$$\nabla_{\tilde{\mathbf{x}}_0}\mathcal{J}^{\text{POD}} = -\mathbf{V}^T\mathbf{B}_0^{-1}(\mathbf{x}_0^{\text{b}} - \mathbf{V}\tilde{\mathbf{x}}_0) - \tilde{\boldsymbol{\lambda}}_0 = 0.$$

The operators $\widehat{\mathbf{H}}_k,\ k = 0, .., m$ and $\widehat{\mathbf{M}}_{t_k\to t_{k+1}},\ k = 0, .., m$ are the Jacobians of the high-fidelity observation operator $\mathcal{H}_k$ and model $\mathcal{M}_{t_k\to t_{k+1}}$ evaluated at $V\tilde{\mathbf{x}}_k$.

4.4 Reduced-Order HMC Sampling Smoothers

One of the key features of HMC sampler is the clever exploration of the state space with guidance based on the distribution geometry. For this the HMC sampling smoother requires not only forward model propagation to evaluate the likelihood term, but also the evaluation of the gradient of the negative-log of the posterior PDF using the adjoint model. This gradient can be approximated using information from the reduced space as obtained from the reduced order model (4.13b).

The HMC algorithm requires that both the momentum $\mathbf{p}$ and the target state $\mathbf{x}$ are vectors of the same dimension. There are two ways to achieve this while using reduced order information:

i) sample the reduced-order subspace only, i.e., collect samples from (4.15a), or

ii) sample the full space, i.e., collect samples from (4.17a) but use an approximate gradient of the posterior negative-log likelihood function obtained in the reduced space.

We next discuss each of these options in detail.

4.4.1 Sampling in the reduced-order space

In this approach the model states are fully projected in the reduced space, $\tilde{\mathbf{x}} \in \mathbb{R}^{\text{NRED}}$. The target posterior distribution, and the potential energy are given by:

$$\begin{aligned}
&\widetilde{\pi}(\tilde{\mathbf{x}}_0) \propto \exp\left(-\widetilde{\mathbf{J}}(\tilde{\mathbf{x}}_0)\right),\\
&\widetilde{\mathbf{J}}(\tilde{\mathbf{x}}_0) = \frac{1}{2}\|\tilde{\mathbf{x}}_0 - \mathbf{V}^T\mathbf{x}_0^{\text{b}}\|^2_{(\mathbf{V}^T\mathbf{B}_0\mathbf{V})^{-1}} + \frac{1}{2}\sum_{k=0}^{m}\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{V}\tilde{\mathbf{x}}_k)\|^2_{\mathbf{R}_k^{-1}},\\
&\tilde{\mathbf{x}}_k = \widetilde{\mathcal{M}}_{k-1,k}(\tilde{\mathbf{x}}_{k-1}), \quad k = 1, 0, \ldots, m; \quad \tilde{\mathbf{x}}_0 = \mathbf{V}^T\mathbf{x}_0,
\end{aligned} \tag{4.15a}$$

and the gradient of the potential energy reads:

$$\nabla_{\tilde{\mathbf{x}}_0}\widetilde{\mathbf{J}}(\tilde{\mathbf{x}}_0) = -\left(\mathbf{V}^T\mathbf{B}_0\mathbf{V}\right)^{-1}\left(\mathbf{V}^T\mathbf{x}_0^{\text{b}} - \tilde{\mathbf{x}}_0\right) - \tilde{\boldsymbol{\lambda}}_0, \tag{4.15b}$$

where $\tilde{\boldsymbol{\lambda}}_0$ is the solution of the ARRA reduced adjoint model (4.14c).

The momentum variable is defined in the reduced space $\tilde{\mathbf{p}}_0 \in \mathbb{R}^{\text{NRED}}$, and the Hamiltonian reads:

$$\widetilde{H}(\tilde{\mathbf{p}}_0, \tilde{\mathbf{x}}_0) = \frac{1}{2}\,\tilde{\mathbf{p}}_0^T\,\widetilde{\mathbf{M}}^{-1}\,\tilde{\mathbf{p}}_0 + \widetilde{\mathbf{J}}(\tilde{\mathbf{x}}_0). \tag{4.16}$$

Following [70, 101], the mass matrix can be chosen as the diagonal matrix $\widetilde{\mathbf{M}} = \text{diag}(\mathbf{V}^T\mathbf{B}_0\mathbf{V})^{-1}$. Note that no further approximations are introduced to the numerical flow produced by the symplectic integrator because all calculations involving models states are calculated in the reduced space.

4.4.2 Sampling in the full space using approximate gradients

Here the model states are initially in the projected subspace defined by $\text{P}_{\text{v}} = \mathbf{V}\mathbf{V}^T$ while the momentum is kept in the full space. The hope is that since the synthetic momentum

is drawn at random from the full space for each proposed state, the symplectic integrator will help the sampler jump between slices of the full space rather that sampling a single subspace, leading to a better ensemble of states obtained from the original target posterior.

The target posterior distribution and the potential energy and its gradient are given by

$$\begin{aligned}
\widehat{\pi}(\mathbf{x}_0) &= \exp\big(-\widehat{\mathbf{J}}(\mathbf{x}_0)\big)\,,\\
\widehat{\mathbf{J}}(\mathbf{x}_0) &= \frac{1}{2}\|\widehat{\mathbf{x}}_0-\mathbf{x}_0^{\mathrm{b}}\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m}\|\mathbf{y}_k-\mathcal{H}_k(\widehat{\mathbf{x}}_k)\|^2_{\mathbf{R}_k^{-1}}\,, \qquad (4.17a)\\
\widehat{\mathbf{x}}_k &= \mathbf{V}\widetilde{\mathcal{M}}_{k-1,k}(\mathbf{V}^T\widehat{\mathbf{x}}_{k-1})\,, \quad k=1,2,\ldots,m; \quad \widehat{\mathbf{x}}_0=\mathbf{x}_0.
\end{aligned}$$

with gradient of the potential energy given by

$$\nabla_{\mathbf{x}_0}\widehat{\mathbf{J}}(\mathbf{x}_0) = -\mathbf{B}_0^{-1}(\mathbf{x}_0^{\mathrm{b}}-\mathbf{x}_0)-\widehat{\boldsymbol{\lambda}}_0, \qquad (4.17b)$$

where $\widehat{\boldsymbol{\lambda}}_0$ is the solution of the following adjoint model

$$\begin{aligned}
\widehat{\boldsymbol{\lambda}}_m &= \mathbf{H}_m^T\mathbf{R}_m^{-1}(\mathbf{y}_m-\mathcal{H}_m(\widehat{\mathbf{x}}_m)), \qquad (4.18a)\\
\widehat{\boldsymbol{\lambda}}_{k-1} &= \mathbf{V}\widetilde{\mathbf{M}}_{k-1,k}^T V^T\widehat{\boldsymbol{\lambda}}_k + \mathbf{H}_{k-1}^T\mathbf{R}_{k-1}^{-1}(\mathbf{y}_{k-1}-\mathcal{H}_{k-1}(\widehat{\mathbf{x}}_{k-1})), \quad k=m,..,1.
\end{aligned}$$

Here $\mathbf{H}_k$ represents the observation operator Jacobian linearized at $\widehat{\mathbf{x}}_k$, $k=0,..,m$, and $\widetilde{\mathbf{M}}_{k-1,k}$ is the Jacobian of the reduced order model evaluated at $V^T\widehat{\mathbf{x}}_k$, $k=0,..,m$. The Hamiltonian in this case takes the form:

$$\widehat{H}(\mathbf{p}_0,\,\mathbf{x}_0) = \frac{1}{2}\,\mathbf{p}_0^T\mathbf{M}^{-1}\mathbf{p}_0 + \widehat{\mathbf{J}}(\mathbf{x}_0)\,. \qquad (4.19)$$

An additional approximation is introduced to the numerical flow produced by the symplectic integrator by the approximation of the gradient of the potential energy. This may require more attention to be paid to the process of parameter tuning especially in the case of very high dimensional spaces.

Algorithm 6 summarizes the sampling process that yields an ensemble of states $\{\tilde{\mathbf{x}}_0(e)\in\mathbb{R}^{\mathrm{N_{RED}}}\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$ in the reduced space, or ensemble of states $\{\widehat{\mathbf{x}}_0(e),\in\mathbb{R}^{\mathrm{N_{state}}}\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$ sampled from the high-fidelity state space with approximate gradient information, respectively

Note that in Algorithm 6, $\mathbf{x}_0^{(i)}$ refers to the model state at the initial time of the assimilation window (or models initial conditions) generated in step i of the Markov chain.

Reduced order smoothing by HMC sampling versus reduced order 4D-Var: The solution of the reduced data assimilation problem (4.14) results in a single reduced order

Algorithm 6 HMC Sampling [101].

1: Initialize the mass matrix: $\widetilde{\mathbf{M}} \in \mathbf{R}^{\mathrm{N_{RED}}\times\mathrm{N_{RED}}}$ for sampling from (4.15a), and $\mathbf{M} \in \mathbf{R}^{\mathrm{N_{state}}\times\mathrm{N_{state}}}$ for sampling from (4.17a).

2: Initialize the chain. Preferably, the initial pair should be as close as possible to the target distribution.

3: At each step i of the Markov chain draw a random auxiliary momentum: $\tilde{\mathbf{p}}_0^{(i)} \sim \mathcal{N}(\mathbf{0}_{\mathrm{N_{RED}}}, \widetilde{\mathbf{M}})$ for sampling from (4.15a), and $\mathbf{p}_0^{(i)} \sim \mathcal{N}(\mathbf{0}_{\mathrm{N_{RED}}}, \mathbf{M})$ for sampling from (4.17a).

4: Use a symplectic numerical integrator (e.g., position Verlet) to advance the current state by a pseudo-time increment T to obtain a *proposal* state :

$$\begin{aligned} &\text{For sampling from (4.15a): } (\tilde{\mathbf{p}}_0^*, \tilde{\mathbf{x}}_0^*) = \widetilde{\phi}_T(\tilde{\mathbf{p}}_0^{(i)}, \tilde{\mathbf{x}}_0^{(i)}). \\ &\text{For sampling from (4.17a): } (\mathbf{p}_0^*, \mathbf{x}_0^*) = \widehat{\phi}_T(\mathbf{p}_0^{(i)}, \mathbf{x}_0^{(i)}), \end{aligned} \tag{4.20}$$

where $\widehat{\Phi}_T$ indicates the flow approximation resulting from approximation of the gradient of the potential energy.

5: For sampling from (4.15a), use the Hamiltonian (4.16) to evaluate the loss of energy' $\Delta\widetilde{H} = \widetilde{H}(\tilde{\mathbf{p}}_0^*, \tilde{\mathbf{x}}_0^*) - \widetilde{H}(\tilde{\mathbf{p}}_0^{(i)}, \tilde{\mathbf{x}}_0^{(i)})$. For sampling (4.17a), use the Hamiltonian (4.19) to approximate the energy loss $\Delta\widehat{H} = \widehat{H}(\mathbf{p}_0^*, \mathbf{x}_0^*) - \widehat{H}(\mathbf{p}_0^{(i)}, \mathbf{x}_0^{(i)})$.

6: Calculate the acceptance probability:

$$\begin{aligned} &\text{For sampling from (4.15a): } a^{(i)} = 1 \wedge e^{-\Delta\widetilde{H}}, \\ &\text{For sampling from (4.17a): } a^{(i)} = 1 \wedge e^{-\Delta\widehat{H}}. \end{aligned} \tag{4.21}$$

7: Discard both current and proposed momentum.

8: **(Acceptance/Rejection)** Draw a uniform random variable $u^{(i)} \sim \mathcal{U}(0,1)$:

i- If $a^{(i)} > u^{(i)}$ accept the proposal as the next sample;

ii- If $a^{(i)} \leq u^{(i)}$ reject the proposal and continue with the current state;

9: Repeat steps 2 to 7 until $\mathrm{N_{ens}}$ distinct samples are drawn.

10: Project the ensemble to the full space.

estimate, namely a reduced order approximation of the analysis state of the dynamical system of concern. The reduced order analysis originates in the reduced space, and is projected to the high fidelity space to give an approximation of the high-fidelity analysis state. Similar to 4D-Var, reduced order 4D-Var does not produce a consistent description of the uncertainty associated with the reduced order analysis.

The reduced order HMC smoothing algorithms proposed here are mainly designed to sample the posterior distribution of the system state at the initial time of an assimilation window. The ensemble points generated by the two versions of the reduced order HMC sampling smoother can be used not only to produce a reduced order analysis approximation, but also to generated approximations of higher order moments such as a flow-dependent analysis error covariance matrix at the initial time of the current assimilation window. The analysis ensemble propagated to the initial time of the next window can also be used to generate a flow-dependent background error covariance matrix.

Moreover, keeping the momentum in the high-fidelity space by sampling (4.17) is expected to result in an approximation that is more accurate than both reduced order 4D-Var and the reduced order HMC smoother by sampling (4.15).

4.5 Properties of the Distributions Sampled with Reduced-Order Models

As explained above, our main goal in this work is to explore the possibility of lowering the computational expense posed by the original HMC smoother [101] by following a reduced-order modeling approach. In the previous Section 4.4, we mentioned that the use of HMC sampling smoother with reduced order models requires following either of two alternatives, namely sampling the posterior distribution fully projected in the lower dimensional subspace, or sampling the high fidelity distribution with gradients approximated using information obtained from the reduced space. In both cases, some amount of information will be lost due to either projecting the posterior PDF, or approximating the components appearing in the likelihood term. More specifically, in the latter case, approximating the negative-log likelihood terms can lead to samples collected from a totally different distribution than the true posterior distribution. In the rest of this section, we discuss the properties of the probability distributions resulting from projection or due to approximation of the negative-log likelihood terms making use of information coming only from a reduced-order subspace.

In the direct case where the posterior distribution is fully projected to the lower dimensional subspace, little can be said about the resulting distribution unless if the true posterior is Gaussian. We explore this case in details in what follows.

4.5.1 Projection of the posterior distribution for linear model and observation operators

In this case the full distribution is projected into the lower-dimensional subspace by approximating both background and observation terms in Equation (4.2b). This projection leads to ensembles generated only in the reduced-space, and are then projected back to the high-fidelity space by left multiplication with $\mathbf{V}$. Projecting the ensembles back to the full space will not change their mass distribution in the case of a linear model and observation operators, and will just embed the ensembles in the full space.

If both the model and the observation operator are linear operators, the posterior (4.2) is a Gaussian distribution $\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0^{\mathrm{a}}, \mathbf{A}_0)$, with a posterior (analysis) mean $\mathbf{x}_0^{\mathrm{a}}$, and an analysis error covariance matrix $\mathbf{A}_0$, i.e.

$$\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0) = \frac{(2\pi)^{\frac{-\mathrm{N_{state}}}{2}}}{\sqrt{|\det(\mathbf{A}_0)|}} \exp\left(-\frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}_0^{\mathrm{a}}\|^2_{\mathbf{A_0}^{-1}}\right). \tag{4.22}$$

The mean and the covariance matrix of the Gaussian posterior (4.22) are given by

$$\begin{aligned} \mathbf{A}_0^{-1} &= \mathbf{B}_0^{-1} + \sum_{k=0}^{m} \mathbf{M}_{0,k}^T \, \mathbf{H}_k^T \, \mathbf{R}_k^{-1} \, \mathbf{H}_k \, \mathbf{M}_{0,k} \,, \\ \mathbf{x}_0^{\mathrm{a}} &= \mathbf{A}_0 \cdot \left(\mathbf{B}_0^{-1} \, \mathbf{x}_0^{\mathrm{b}} + \sum_{k=0}^{m} \mathbf{M}_{0,k}^T \, \mathbf{H}_k^T \, \mathbf{R}_k^{-1} \, \mathbf{y}_k \right). \end{aligned} \tag{4.23}$$

Projecting this PDF onto the subspace spanned by columns of the matrix $\mathbf{V}$ (e.g., POD basis) results in a projected PDF $\widetilde{\mathcal{P}}^{\mathrm{a}}(\tilde{\mathbf{x}}_0) = \mathcal{N}(\mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}, \mathbf{V}^T\mathbf{A}_0\mathbf{V})$; $\tilde{\mathbf{x}}_0 \in \mathbb{R}^{\mathrm{N_{RED}}}$, i.e.,

$$\widetilde{\mathcal{P}}^{\mathrm{a}}(\tilde{\mathbf{x}}_0) = \frac{(2\pi)^{\frac{-\mathrm{N_{RED}}}{2}}}{\sqrt{|\det(\mathbf{V}^T\mathbf{A}_0\mathbf{V})|}} \exp\left(-\frac{1}{2}\|\tilde{\mathbf{x}}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\|^2_{(\mathbf{V^T A_0 V})^{-1}}\right). \tag{4.24}$$

The linear transformation, of the analysis state, with the orthogonal projector $\mathrm{P_v} = \mathbf{V}\mathbf{V}^T$, results as well in the Gaussian distribution $\widehat{\mathcal{P}}^{\mathrm{a}}(\widehat{\mathbf{x}}_0) = \mathcal{N}(\mathrm{P_v}\mathbf{x}_0^{\mathrm{a}}, \mathrm{P_v}\mathbf{A}_0\mathrm{P_v}) \equiv \mathcal{N}(\widehat{\mathbf{x}}_0^{\mathrm{a}}, \widehat{\mathbf{A}}_0)$, $\widehat{\mathbf{x}}_0 \in \mathbb{R}^{\mathrm{N_{state}}}$. The covariance matrix $\widehat{\mathbf{A}}_0$ however is not full rank, and the Gaussian distribution is degenerate. The density function of this singular distribution can be rigorously

formulated by defining a restriction of Lebesgue measure to the affine subspace of $\mathbb{R}^{\mathrm{N_{state}}}$ whose dimension is limited to $\mathrm{rank}(\widehat{\mathbf{A}}_0)$. The Gaussian (singular) density then formula takes the form [143, 144]

$$\widehat{\mathcal{P}}^{\mathrm{a}}(\widehat{\mathbf{x}}_0) = \frac{(2\pi)^{\frac{-\mathrm{N_{RED}}}{2}}}{\sqrt{|\det^*(\widehat{\mathbf{A}}_0)|}} \exp\left(-\frac{1}{2}\|\widehat{\mathbf{x}}_0 - \widehat{\mathbf{x}}_0^{\mathrm{a}}\|^2_{\widehat{\mathbf{A}}_0^\dagger}\right) \cdot \delta_{(\mathbf{I}-\mathrm{P_v})\widehat{\mathbf{x}}_0} \tag{4.25}$$

where $\det^*$ is the pseudo determinant, and $\dagger$ refers to the matrix pseudo inverse. Of course, $\tilde{\mathbf{x}}_0 \in \mathbb{R}^{\mathrm{N_{RED}}}$, $\widehat{\mathbf{x}}_0 \in \mathbb{R}^{\mathrm{N_{state}}}$. One can think of the PDF (4.25) as a version of (4.24) embedded in the high-fidelity state space.

Theorem 1. *If $\widetilde{\mathcal{P}}^{\mathrm{a}}(\tilde{\mathbf{x}}_0)$ and $\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{V}\tilde{\mathbf{x}}_0)$ are the distributions defined in* (4.24) *and* (4.25)*, respectively, then the following result holds true for a given reduced basis* $\mathbf{V}$

$$\widetilde{\mathcal{P}}^{\mathrm{a}}(\tilde{\mathbf{x}}_0) = \widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{V}\tilde{\mathbf{x}}_0),\ \forall \tilde{\mathbf{x}}_0 \in \mathbb{R}^{\mathrm{N_{RED}}}. \tag{4.26}$$

Proof. For this purpose it is sufficient to prove that

$$\|\widehat{\mathbf{x}}_0 - \widehat{\mathbf{x}}_0^{\mathrm{a}}\|^2_{\widehat{\mathbf{A}}_0^\dagger} = \|\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\|^2_{\left(\mathbf{V}^T\mathbf{A_0}\mathbf{V}\right)^{-1}} \tag{4.27}$$

Assume the relation given by Equation (4.27) is correct, we get the following:

$$\begin{aligned}
\|\widehat{\mathbf{x}}_0 - \widehat{\mathbf{x}}_0^{\mathrm{a}}\|^2_{\widehat{\mathbf{A}}_0^\dagger} &= \|\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\|^2_{\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}}, \\
(\widehat{\mathbf{x}}_0 - \widehat{\mathbf{x}}_0^{\mathrm{a}})^T\widehat{\mathbf{A}}_0^\dagger(\widehat{\mathbf{x}}_0 - \widehat{\mathbf{x}}_0^{\mathrm{a}}) &= (\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}})^T\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}) \\
(\mathrm{P_v}\mathbf{x}_0 - \mathrm{P_v}\mathbf{x}_0^{\mathrm{a}})^T\widehat{\mathbf{A}}_0^\dagger(\mathrm{P_v}\mathbf{x}_0 - \mathrm{P_v}\mathbf{x}_0^{\mathrm{a}}) &= (\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}})^T\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}})
\end{aligned} \tag{4.28a}$$

Or equivalently:

$$\begin{aligned}
0 &= (\mathrm{P_v}\mathbf{x}_0 - \mathrm{P_v}\mathbf{x}_0^{\mathrm{a}})^T\,\widehat{\mathbf{A}}_0^\dagger\,(\mathrm{P_v}\mathbf{x}_0 - \mathrm{P_v}\mathbf{x}_0^{\mathrm{a}}) - (\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}})^T\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}) \\
&= \left(\mathbf{V}(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}})\right)^T\,\widehat{\mathbf{A}}_0^\dagger\,\left(\mathbf{V}(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}})\right) - \left(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\right)^T\,\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}\,\left(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\right) \\
&= \left(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\right)^T\,\mathbf{V}^T\widehat{\mathbf{A}}_0^\dagger\mathbf{V}\,\left(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\right) - \left(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\right)^T\,\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}\,\left(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}\right) \\
&= (\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}})^T\,\left(\mathbf{V}^T\widehat{\mathbf{A}}_0^\dagger\mathbf{V} - \left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}\right)(\mathbf{V}^T\mathbf{x}_0 - \mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}).
\end{aligned} \tag{4.28b}$$

This holds true if the matrix $\mathbf{V}^T\widehat{\mathbf{A}}_0^\dagger\mathbf{V} - \left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}$ is equal to a zero matrix. The matrix $\mathbf{V}$ has orthonormal columns and consequently $\left(\mathrm{P_v}\mathbf{A}_0\mathrm{P_v}\right)^\dagger = \left(\mathbf{V}\mathbf{V}^T\mathbf{A}_0\mathrm{P_v}\right)^\dagger = \left(\mathbf{V}^T\mathbf{A}_0\mathrm{P_v}\right)^\dagger\mathbf{V}^\dagger$. Since the pseudo inverse and the transpose operations are commutative, we get the following:

$$\begin{aligned}
\left(\mathbf{V}^T\mathbf{A}_0\mathrm{P_v}\right)^\dagger &= \left(\left(\mathrm{P_v}\mathbf{A}_0^T\mathbf{V}\right)^T\right)^\dagger, \\
&= \left(\mathbf{V}^T\right)^\dagger\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^\dagger,
\end{aligned} \tag{4.28c}$$

and consequently:

$$\begin{aligned}\mathbf{V}^T\widehat{\mathbf{A}}_0^\dagger\mathbf{V} = \mathbf{V}^T(\mathrm{P_v}\mathbf{A}_0\mathrm{P_v})^\dagger\mathbf{V} &= \mathbf{V}^T\left(\mathbf{V}^T\right)^\dagger\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^\dagger\mathbf{V}^\dagger\mathbf{V}\,,\\ &= \mathbf{V}^T\left(\mathbf{V}^T\right)^\dagger\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}\mathbf{V}^\dagger\mathbf{V}\,,\\ &= (\mathbf{V}^T\mathbf{V})\left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}(\mathbf{V}^T\mathbf{V})\,,\\ &= \left(\mathbf{V}^T\mathbf{A}_0\mathbf{V}\right)^{-1}\,,\end{aligned} \tag{4.28d}$$

where $\mathbf{V}^\dagger = \mathbf{V}^T$, and $\left(\mathbf{V}^T\right)^\dagger = \mathbf{V}$ since $\mathbf{V}$ has orthonormal columns. This means that the relation (4.27) holds, and the equivalence between (4.24) and (4.25) follows immediately. □

This result suggests that sampling from the distribution (4.25) can be carried out efficiently by sampling the distribution (4.24), then projecting the ensembles back to the full space using $\mathbf{V}$.

By determining the Kullback Leibler (KL) [145] divergence measure between the high fidelity distribution $\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)$ and the probability distribution $\widehat{\mathcal{P}}^{\mathrm{a}}(\widehat{\mathbf{x}}_0)$, one can estimate the error between the projected samples obtained using distribution (4.24) and those sampled from the high fidelity distribution $\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)$.

Theorem 2. *The KL divergence measure between the Gaussian distribution* $\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0)$ *given by* (4.25)*, and the probability distribution* $\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)$ *defined in* (4.22)*, is given as*

$$\begin{aligned}D_{\mathrm{KL}}\left(\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0)\|\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)\right) = \frac{1}{2}(\mathrm{N_{state}}-\mathrm{N_{RED}})\ln(2\pi) + \frac{1}{2}\ln\left(\frac{|\det(\mathbf{A}_0)|}{|\det^*(\widehat{\mathbf{A}}_0)|}\right)\\ + \frac{1}{2}\|\widehat{\mathbf{x}}_0^{\mathrm{a}}-\mathbf{x}_0^{\mathrm{a}}\|_{\mathbf{A}_0^{-1}} + \frac{1}{2}\mathrm{Tr}\left(\left(\mathbf{A}_0^{-1}-\widehat{\mathbf{A}}_0^\dagger\right)\widehat{\mathbf{A}}_0\right)\,,\end{aligned} \tag{4.29}$$

where $\mathbf{V}\in\mathbf{R}^{\mathrm{N_{state}}\times\mathrm{N_{RED}}}$ *and* $\mathrm{N_{RED}} < \mathrm{N_{state}}$.

Proof. The KL measure is obtained as

$$D_{\mathrm{KL}}\left(\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0)\|\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)\right) = \mathrm{E}_{\widehat{\mathcal{P}}^{\mathrm{a}}}\left[\ln\left(\frac{\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0)}{\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)}\right)\right], \tag{4.30a}$$

$$\begin{aligned}\ln\left(\frac{\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0)}{\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)}\right) &= \ln\left(\frac{(2\pi)^{\frac{-\mathrm{N_{RED}}}{2}}}{\sqrt{|\det^*(\widehat{\mathbf{A}}_0)|}}\right) + \ln\left(\frac{\sqrt{|\det(\mathbf{A}_0)|}}{(2\pi)^{\frac{-\mathrm{N_{state}}}{2}}}\right) \\ &\quad + \ln\left(\frac{\exp\left(-\frac{1}{2}\|\mathbf{x}_0-\widehat{\mathbf{x}}_0^{\mathrm{a}}\|^2_{\widehat{\mathbf{A}}_0^{\dagger}}\right)}{\exp\left(-\frac{1}{2}\|\mathbf{x}_0-\mathbf{x}_0^{\mathrm{a}}\|^2_{\mathbf{A_0}^{-1}}\right)}\right)\end{aligned} \tag{4.30b}$$

$$\begin{aligned}&= \frac{(\mathrm{N_{state}}-\mathrm{N_{RED}})\ln(2\pi)}{2} + \frac{1}{2}\ln\left(\frac{|\det(\mathbf{A}_0)|}{|\det^*(\widehat{\mathbf{A}}_0)|}\right) \\ &\quad + \frac{1}{2}\left(\|\mathbf{x}_0-\mathbf{x}_0^{\mathrm{a}}\|^2_{\mathbf{A_0}^{-1}} - \|\mathbf{x}_0-\widehat{\mathbf{x}}_0^{\mathrm{a}}\|^2_{\widehat{\mathbf{A}}_0^{\dagger}}\right)\end{aligned} \tag{4.30c}$$

$$\begin{aligned}\mathrm{E}_{\widehat{\mathcal{P}}^{\mathrm{a}}}\left[\ln\left(\frac{\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0)}{\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)}\right)\right] &= \frac{(\mathrm{N_{state}}-\mathrm{N_{RED}})\ln(2\pi)}{2} + \frac{1}{2}\ln\left(\frac{|\det(\mathbf{A}_0)|}{|\det^*(\widehat{\mathbf{A}}_0)|}\right) \\ &\quad + \frac{1}{2}\mathrm{E}_{\widehat{\mathcal{P}}^{\mathrm{a}}}\left[\left(\|\mathbf{x}_0-\mathbf{x}_0^{\mathrm{a}}\|^2_{\mathbf{A_0}^{-1}} - \|\mathbf{x}_0-\widehat{\mathbf{x}}_0^{\mathrm{a}}\|^2_{\widehat{\mathbf{A}}_0^{\dagger}}\right)\right],\end{aligned} \tag{4.30d}$$

where $\ln\left(\frac{|\det(\mathbf{A}_0)|}{|\det^*(\widehat{\mathbf{A}}_0)|}\right)$ is the sum of logarithms of eigenvalues of $\mathbf{A}_0$ lost due to projection. This value can be also replaced with $ln\left(\frac{|\det(\mathbf{A}_0)|}{|\det(\mathbf{V}^T\mathbf{A}_0\mathbf{V})|}\right)$ due to the nature of the matrix $\mathbf{V}$. The expectation of the quadratic terms in Equation (4.30d) can be obtained as follows:

$$\mathrm{E}_{\widehat{\mathcal{P}}^{\mathrm{a}}}\left[\left(\|\mathbf{x}_0-\mathbf{x}_0^{\mathrm{a}}\|^2_{\mathbf{A}_0{}^{-1}} - \|\mathbf{x}_0-\widehat{\mathbf{x}}_0^{\mathrm{a}}\|^2_{\widehat{\mathbf{A}}_0^{\dagger}}\right)\right] = \mathrm{E}_{\widehat{\mathcal{P}}^{\mathrm{a}}}\left[\left(\|\mathbf{x}_0-\mathbf{x}_0^{\mathrm{a}}\|^2_{\mathbf{A}_0{}^{-1}}\right] - \mathrm{E}_{\widehat{\mathcal{P}}^{\mathrm{a}}}\left[\|\mathbf{x}_0-\widehat{\mathbf{x}}_0^{\mathrm{a}}\|^2_{\widehat{\mathbf{A}}_0^{\dagger}}\right)\right] \tag{4.31a}$$

$$\begin{aligned}&= \mathrm{E}_{\widehat{\mathcal{P}}^{\mathrm{a}}}\left[(\mathbf{x}_0-\mathbf{x}_0^{\mathrm{a}})^T\mathbf{A}_0{}^{-1}(\mathbf{x}_0-\mathbf{x}_0^{\mathrm{a}})\right] \\ &\quad - \mathrm{E}_{\widehat{\mathcal{P}}^{\mathrm{a}}}\left[(\mathbf{x}_0-\widehat{\mathbf{x}}_0^{\mathrm{a}})^T\widehat{\mathbf{A}}_0^{\dagger}(\mathbf{x}_0-\widehat{\mathbf{x}}_0^{\mathrm{a}})\right]\end{aligned} \tag{4.31b}$$

$$\begin{aligned}&= (\widehat{\mathbf{x}}_0^{\mathrm{a}}-\mathbf{x}_0^{\mathrm{a}})^T\mathbf{A}_0^{-1}(\widehat{\mathbf{x}}_0^{\mathrm{a}}-\mathbf{x}_0^{\mathrm{a}}) \\ &\quad + \mathrm{Tr}\left(\mathbf{A}_0^{-1}\,\widehat{\mathbf{A}}_0\right) - \mathrm{Tr}\left(\widehat{\mathbf{A}}_0^{\dagger}\,\widehat{\mathbf{A}}_0\right).\end{aligned} \tag{4.31c}$$

From Equations (4.31), and (4.30), we obtain:

$$D_{\mathrm{KL}}\left(\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0)\|\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)\right) = \frac{(\mathrm{N_{state}} - \mathrm{N_{RED}})\ln(2\pi)}{2} + \frac{1}{2}\ln\left(\frac{|\det(\mathbf{A}_0)|}{|\det^*(\widehat{\mathbf{A}}_0)|}\right) + \frac{1}{2}\|\widehat{\mathbf{x}}_0^{\mathrm{a}} - \mathbf{x}_0^{\mathrm{a}}\|_{\mathbf{A}_0^{-1}} \tag{4.32a}$$

$$+ \frac{1}{2}\mathrm{Tr}\left(\mathbf{A}_0^{-1}\,\widehat{\mathbf{A}}_0\right) - \frac{1}{2}\mathrm{Tr}\left(\widehat{\mathbf{A}}_0^{\dagger}\,\widehat{\mathbf{A}}_0\right)$$

$$= \frac{(\mathrm{N_{state}} - \mathrm{N_{RED}})\ln(2\pi)}{2} + \frac{1}{2}\ln\left(\frac{|\det(\mathbf{A}_0)|}{|\det^*(\widehat{\mathbf{A}}_0)|}\right) + \frac{1}{2}\|\widehat{\mathbf{x}}_0^{\mathrm{a}} - \mathbf{x}_0^{\mathrm{a}}\|_{\mathbf{A}_0^{-1}} \tag{4.32b}$$

$$+ \frac{1}{2}\mathrm{Tr}\left(\mathbf{A}_0^{-1}\,\widehat{\mathbf{A}}_0 - \widehat{\mathbf{A}}_0^{\dagger}\,\widehat{\mathbf{A}}_0\right)$$

$$= \frac{1}{2}(\mathrm{N_{state}} - \mathrm{N_{RED}})\ln(2\pi) + \frac{1}{2}\ln\left(\frac{|\det(\mathbf{A}_0)|}{|\det^*(\widehat{\mathbf{A}}_0)|}\right) + \frac{1}{2}\|\widehat{\mathbf{x}}_0^{\mathrm{a}} - \mathbf{x}_0^{\mathrm{a}}\|_{\mathbf{A}_0^{-1}} \tag{4.32c}$$

$$+ \frac{1}{2}\mathrm{Tr}\left(\left(\mathbf{A}_0^{-1} - \widehat{\mathbf{A}}_0^{\dagger}\right)\,\widehat{\mathbf{A}}_0\right),$$

which completes the proof. □

This measure can be used to quantify the quality of POD basis given an estimation of the analysis error covariance matrix, e.g., based on an ensemble of states, sampled from the high fidelity distribution, or approximated based on statistics of the 4D-Var cost functional. Notice that the KL measure given in (4.29) is finite since $\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0)$ is absolutely continuous with respect to $\mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)$ (and it is zero only if $\widehat{\mathcal{P}}^{\mathrm{a}}(\mathbf{x}_0) = \mathcal{P}^{\mathrm{a}}(\mathbf{x}_0)$). For this reason, we set the projected PDF as the reference density in the KL measure.

4.5.2 Approximating the likelihood function using reduced order models

In the latter approach, the background term is kept in the high fidelity space, while only the terms involving model propagations are approximated using reduced-order models. This means that the target distribution is the PDF give by (4.17a). The use of this approximation in the HMC algorithm results in samples collected from the distribution (4.17a). This approximation maintains the background term in the full space, while the model states involved in the observation term are approximated in the lower-dimensional subspace.

This means that the posterior distribution is non-degenerate in the full space due to the background term. However, it is not immediately obvious which distributions samples will be collected from. In Theorem 3 we show the link between posterior distribution given by (4.17a) and the distribution defined in (4.2).

Theorem 3. *The posterior distribution* $\boldsymbol{\pi}$ *defined in* (4.2) *associated with the high-fidelity model* (4.4) *is proportional to the analysis posterior distribution* $\widehat{\boldsymbol{\pi}}$ *introduced in* (4.17a) *associated with the reduced order model, by the ratio between joint likelihood functions given projected and high-fidelity states, i.e.*

$$\widehat{\boldsymbol{\pi}}(\mathbf{x}_0) = \boldsymbol{\pi}(\mathbf{x}_0) \cdot \prod_{k=1}^{m} \frac{\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k = \mathbf{V}\,\widetilde{\mathcal{M}}_{0,k}(\mathbf{V}^T\mathbf{x}_0))}{\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k = \mathcal{M}_{0,k}(\mathbf{x}_0))}\,. \tag{4.33}$$

Proof. The exact and the approximate posterior distributions $\boldsymbol{\pi}(\mathbf{x}_0)$, $\widehat{\boldsymbol{\pi}}(\mathbf{x}_0)$ are generally described as follows

$$\begin{aligned}\boldsymbol{\pi}(\mathbf{x}_0) = \mathcal{P}^{\mathrm{a}}(\mathbf{x}_0) &= \mathcal{P}^{\mathrm{b}}(\mathbf{x}_0) \cdot \mathcal{P}(\mathbf{y}_0|\mathbf{x}_0) \cdot \prod_{k=1}^{m} \mathcal{P}(\mathbf{y}_k|\mathbf{x}_k) \cdot \mathcal{P}(\mathbf{x}_k|\mathbf{x}_{k-1}) \\ &= \mathcal{P}^{\mathrm{b}}(\mathbf{x}_0) \cdot \mathcal{P}(\mathbf{y}_0|\mathbf{x}_0) \cdot \prod_{k=1}^{m} \mathcal{P}(\mathbf{y}_k|\mathbf{x}_k = \mathcal{M}_{0,k}(\mathbf{x}_0))\,,\end{aligned} \tag{4.34a}$$

$$\begin{aligned}\widehat{\boldsymbol{\pi}}(\mathbf{x}_0) &= \mathcal{P}^{\mathrm{b}}(\mathbf{x}_0) \cdot \mathcal{P}(\mathbf{y}_0|\widehat{\mathbf{x}}_0) \cdot \prod_{k=1}^{m} \mathcal{P}(\mathbf{y}_k|\widehat{\mathbf{x}}_k) \cdot \mathcal{P}(\widehat{\mathbf{x}}_k|\widehat{\mathbf{x}}_{k-1}) \\ &= \mathcal{P}^{\mathrm{b}}(\mathbf{x}_0) \cdot \mathcal{P}(\mathbf{y}_0|\mathbf{x}_0) \cdot \prod_{k=1}^{m} \mathcal{P}(\mathbf{y}_k|\mathbf{x}_k = \mathbf{V}\,\widetilde{\mathcal{M}}_{0,k}(\mathbf{V}^T\mathbf{x}_0))\,.\end{aligned} \tag{4.34b}$$

This leads to the following:

$$\begin{aligned}\frac{\widehat{\boldsymbol{\pi}}(\mathbf{x}_0)}{\boldsymbol{\pi}(\mathbf{x}_0)} &= \frac{\prod_{k=1}^{m} \mathcal{P}(\mathbf{y}_k|\mathbf{x}_k = \mathbf{V}\,\widetilde{\mathcal{M}}_{0,k}(\mathbf{V}^T\mathbf{x}_0))}{\prod_{k=1}^{m} \mathcal{P}(\mathbf{y}_k|\mathbf{x}_k = \mathcal{M}_{0,k}(\mathbf{x}_0))}\,, \\ \widehat{\boldsymbol{\pi}}(\mathbf{x}_0) &= \boldsymbol{\pi}(\mathbf{x}_0) \cdot \prod_{k=1}^{m} \frac{\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k = \mathbf{V}\,\widetilde{\mathcal{M}}_{0,k}(\mathbf{V}^T\mathbf{x}_0))}{\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k = \mathcal{M}_{0,k}(\mathbf{x}_0))}\,.\end{aligned} \tag{4.34c}$$

□

This result suggests that the larger the distances $\|\mathbf{x}_k - \mathbf{V}\tilde{x}_k\|_2,\ k = 1, 2, .., m$ are, the more different the distributions $\boldsymbol{\pi}$ and $\widehat{\boldsymbol{\pi}}$ will be. By selecting appropriate reduced manifolds $\mathbf{V}$ and decreasing the error associated with the reduced order models, the ratio can be brought closer to 1.

Corollary 3.1. *The KL divergence measure between the original posterior* (4.2) *and the approximated distribution* (4.17a) *is:*

$$\begin{aligned} D_{\mathrm{KL}}(\widehat{\boldsymbol{\pi}}||\boldsymbol{\pi}) &= \mathrm{E}_{\widehat{\boldsymbol{\pi}}}\left[\ln(\widehat{\boldsymbol{\pi}}) - \ln(\boldsymbol{\pi})\right] = \mathrm{E}_{\widehat{\boldsymbol{\pi}}}\left[\mathcal{J}\mathbf{x}_0) - \widehat{\mathbf{J}}(\mathbf{x}_0)\right] \\ &= \mathrm{E}_{\widehat{\boldsymbol{\pi}}}\left[\mathbf{J}^{\mathrm{obs}}(\mathbf{x}_0) - \widehat{\mathbf{J}}^{\mathrm{obs}}(\mathbf{x}_0)\right] , \end{aligned} \tag{4.35}$$

where $\mathbf{J}^{\mathrm{obs}}(\mathbf{x}_0)$*, and* $\widehat{\mathbf{J}}^{\mathrm{obs}}(\mathbf{x}_0)$ *are the observation terms in the full and the approximate 4D-Var cost function.*

Corollary 3.2. *In the filtering case, where only one observation is assimilated, if the initial condition is projected on the columns of* $\mathbf{V}$ *to approximate the likelihood term, the posterior distribution is given by:*

$$\widehat{\boldsymbol{\pi}}(\mathbf{x}_0) = \widehat{\mathcal{P}^{\mathrm{a}}}(\mathbf{x}_0) \propto \frac{\boldsymbol{\pi}(\mathbf{x}_0^{\parallel}) \cdot \mathcal{P}^{\mathrm{b}}(\mathbf{x}_0)}{\mathcal{P}^{\mathrm{b}}(\mathbf{x}_0^{\parallel})} , \tag{4.36}$$

where $\mathbf{x}_0 = \mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp}$*, with* $\mathbf{x}_0^{\parallel} \in$ *range*$(\mathbf{V})$ *and* $\mathbf{x}_0^{\perp} \in$ *null*$(\mathbf{V}^T)$

Proof. The two states $\mathbf{x}_0$ and $\mathbf{x}_0^{\parallel}$ differ only along a direction $\mathbf{x}_0^{\perp}$ orthogonal to the reduced space, that is $\mathbf{V}^T\mathbf{x}_0^{\perp} = 0$, and consequently

$$\mathbf{V}^T\,\mathbf{x}_0 = \mathbf{V}^T(\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp}) = \mathbf{V}^T\,\mathbf{x}_0^{\parallel}.$$

In the filtering case the cost function reads:

$$\begin{aligned} \widehat{\mathbf{J}}(\mathbf{x}_0) &= \frac{1}{2}||\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp} - \mathbf{x}_0^{\mathrm{b}}||^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\left\|\mathbf{y}_k - \mathcal{H}_0\left(\mathbf{V}\mathbf{V}^T\mathbf{x}_0^{\parallel}\right)\right\|_{\mathbf{R}_0^{-1}} \\ &= \frac{1}{2}||\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp} - \mathbf{x}_0^{\mathrm{b}}||^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\left\|\mathbf{y}_k - \mathcal{H}_0(\mathbf{x}_0^{\parallel})\right\|_{\mathbf{R}_0^{-1}} , \\ \mathbf{J}(\mathbf{x}_0^{\parallel}) &= \frac{1}{2}||\mathbf{x}_0^{\parallel} - \mathbf{x}_0^{\mathrm{b}}||^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\left\|\mathbf{y}_k - \mathcal{H}_0\left(\mathbf{x}_0^{\parallel}\right)\right\|_{\mathbf{R}_0^{-1}}, \\ -\widehat{\mathbf{J}}(\mathbf{x}_0) &= -\mathcal{J}\mathbf{x}_0^{\parallel}) - \frac{1}{2}||\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp} - \mathbf{x}_0^{\mathrm{b}}||^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}||\mathbf{x}_0^{\parallel} - \mathbf{x}_0^{\mathrm{b}}||^2_{\mathbf{B}_0^{-1}}. \end{aligned} \tag{4.37a}$$

Exponentiating of both sides leads to the following:

$$\begin{aligned} \exp\left(-\widehat{\mathbf{J}}(\mathbf{x}_0)\right) &= \exp\left(-\frac{1}{2}||\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp} - \mathbf{x}_0^{\mathrm{b}}||^2_{\mathbf{B}_0^{-1}}\right)\exp\left(-\mathcal{J}\mathbf{x}_0^{\parallel})\right)\exp\left(\frac{1}{2}||\mathbf{x}_0^{\parallel} - \mathbf{x}_0^{\mathrm{b}}||^2_{\mathbf{B}_0^{-1}}\right), \\ \widehat{\boldsymbol{\pi}}(\mathbf{x}_0) &\propto \frac{\boldsymbol{\pi}(\mathbf{x}_0^{\parallel}) \cdot \mathcal{P}^{\mathrm{b}}(\mathbf{x}_0)}{\mathcal{P}^{\mathrm{b}}(\mathbf{x}_0^{\parallel})} . \end{aligned} \tag{4.38}$$

This completes the proof. □

Corollary 3.3. *In the filtering case, if* $\mathbf{x}_0^{\perp} = 0$ *the two distributions* $\widehat{\boldsymbol{\pi}}(\mathbf{x}_0)$*, and* $\boldsymbol{\pi}(\mathbf{x}_0)$ *coincide, and if* $\mathbf{x}_0^{\parallel} = 0$ *then the reduced distribution* $\widehat{\boldsymbol{\pi}}(\mathbf{x}_0)$ *coincides with the background distribution* $\mathcal{P}^{\mathrm{b}}(\mathbf{x}_0)$*.*

In the general case we have:

$$\begin{aligned}\widehat{\mathbf{J}}(\mathbf{x}_0) = \widehat{\mathbf{J}}(\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp}) &= \frac{1}{2}\|\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp} - \mathbf{x}_0^{\mathrm{b}}\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m}\left\|\mathbf{y}_k - \mathcal{H}_k\left(\mathbf{V}\widetilde{\mathcal{M}}_{0,k}\left(\mathbf{V}^T(\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp})\right)\right)\right\|_{\mathbf{R}_k^{-1}} \\ &= \frac{1}{2}\|\mathbf{x}_0^{\parallel} + \mathbf{x}_0^{\perp} - \mathbf{x}_0^{\mathrm{b}}\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m}\left\|\mathbf{y}_k - \mathcal{H}_k\left(\mathbf{V}\widetilde{\mathcal{M}}_{0,k}\left(\mathbf{V}^T\mathbf{x}_0^{\parallel}\right)\right)\right\|_{\mathbf{R}_k^{-1}}, \end{aligned} \tag{4.39a}$$

$$\mathbf{J}(\mathbf{x}_0^{\parallel}) = \frac{1}{2}\|\mathbf{x}_0^{\parallel} - \mathbf{x}_0^{\mathrm{b}}\|^2_{\mathbf{B}_0^{-1}} + \frac{1}{2}\sum_{k=0}^{m}\left\|\mathbf{y}_k - \mathcal{H}_k\left(\mathcal{M}_{0,k}\left(\mathbf{x}_0^{\parallel}\right)\right)\right\|_{\mathbf{R}_k^{-1}}, \tag{4.39b}$$

Corollary 3.4. *The posterior* $\widehat{\boldsymbol{\pi}}$ (4.17a) *is Gaussian with analysis covariance and mean:*

$$\begin{aligned}\widehat{\mathbf{A}}_0^{-1} &= \mathbf{B}_0^{-1} + \sum_{k=0}^{m}\mathbf{V}\,\widetilde{\mathbf{M}}_{0,k}^T\,\mathbf{H}_k^T\,\mathbf{R}_k^{-1}\,\mathbf{H}_k\,\widetilde{\mathbf{M}}_{0,k}\,\mathbf{V}^T \\ \widehat{\mathbf{x}}_0^{\mathrm{a}} &= \widehat{\mathbf{A}}_0\cdot\left(\mathbf{B}_0^{-1}\,\mathbf{x}_0^{\mathrm{b}} + \sum_{k=0}^{m}\mathbf{V}\widetilde{\mathbf{M}}_{0,k}^T\,\mathbf{H}_k^T\,\mathbf{R}_k^{-1}\,\mathbf{y}_k\right).\end{aligned} \tag{4.40}$$

From Equations (4.23) and (4.40) we conclude that the analysis mean and covariance associated with the distribution $\widehat{\boldsymbol{\pi}}$ (4.17a) are not obtained simply by projecting the mean and covariance of the high fidelity distribution $\boldsymbol{\pi}$ (4.2), i.e., $\widehat{\mathbf{A}}_0 \neq \mathbf{V}\mathbf{V}^T\mathbf{A}_0\mathbf{V}\mathbf{V}^T$ and $\widehat{\mathbf{x}}_0^{\mathrm{a}} \neq \mathbf{V}\mathbf{V}^T\mathbf{x}_0^{\mathrm{a}}$.

Corollary 3.5. *For a constant model operator* $\mathbf{M}_{k-1,k} = \mathbf{M}$ *the mean and the covariance of the high-fidelity posterior* (4.2) *are*

$$\begin{aligned}\mathbf{A}_0^{-1} &= \mathbf{B}_0^{-1} + \sum_{k=0}^{m}(\mathbf{M}^k)^T\,\mathbf{H}_k^T\,\mathbf{R}_k^{-1}\,\mathbf{H}_k\,(\mathbf{M}^k), \\ \mathbf{x}_0^{\mathrm{a}} &= \mathbf{A}_0\cdot\left(\mathbf{B}_0^{-1}\,\mathbf{x}_0^{\mathrm{b}} + \sum_{k=0}^{m}(\mathbf{M}^k)^T\,\mathbf{H}_k^T\,\mathbf{R}_k^{-1}\,\mathbf{y}_k\right),\end{aligned} \tag{4.41}$$

while in the case of posterior (4.17a)*, the associated analysis covariance and mean are*

$$\begin{aligned}\widehat{\mathbf{A}}_0^{-1} &= \mathbf{B}_0^{-1} + \sum_{k=0}^{m}\left((\mathrm{P_v}\mathbf{M}\mathrm{P_v})^k\right)^T\,\mathbf{H}_k^T\,\mathbf{R}_k^{-1}\,\mathbf{H}_k\,(\mathrm{P_v}\mathbf{M}\mathrm{P_v})^k \\ \widehat{\mathbf{x}}_0^{\mathrm{a}} &= \widehat{\mathbf{A}}_0\cdot\left(\mathbf{B}_0^{-1}\,\mathbf{x}_0^{\mathrm{b}} + \sum_{k=0}^{m}\left((\mathrm{P_v}\mathbf{M}\mathrm{P_v})^k\right)^T\,\mathbf{H}_k^T\,\mathbf{R}_k^{-1}\,\mathbf{y}_k\right).\end{aligned} \tag{4.42}$$

A closed form for the distribution (4.17a) can be obtained if a) the observation errors are defined given the state vectors in the lower-dimensional subspace embedded in the full space (the projected space), b) the observation errors at time t_k follow Gaussian distribution with zero mean and covariance matrix $\widehat{\mathbf{R}}_k$, that is

$$\widehat{\mathbf{e}}_k^{\text{obs}} = \mathbf{y}_k - \mathcal{H}\mathbf{V}\hat{\mathbf{x}}_k) \sim \mathcal{N}(0, \widehat{\mathbf{R}}_k)\,, \tag{4.43}$$

and c) forcing the regular assumptions of time independence of observation errors, and independence from model background state (in the smaller space).

In this work, we have considered a single reduced order subspace to approximate the model flow on each assimilation cycle. One can build reduced order approximations of the model dynamics in lower dimensional subspaces with different dimensions and add correction terms to the obtained ensemble to further enhance the quality of the collected ensemble states and the computed statistics. This would resemble the idea behind multilevel Monte Carlo simulations [146, 147].

4.6 Numerical Results

In this section we test numerically the reduced order sampling algorithms using the shallow-water equations (SWE) model in Cartesian coordinates.

4.6.1 The SWE model

Many phenomena in fluid dynamics are characterized by horizontal length scale much greater than the vertical length, consequently when equipped with Coriolis forces, the shallow water equations model (SWE) becomes a valuable tool in atmospheric modeling, as a simplification of the primitive equations of atmospheric flow. Their solutions represent many of the types of motion found in the real atmosphere, including slow-moving Rossby waves and fast-moving gravity waves [148]. The alternating direction fully implicit finite difference scheme [149] was considered in this chapter and it is stable for large CFL condition numbers.

The SWE model using the β-plane approximation on a rectangular domain is introduced (see [149, 150])

$$\frac{\partial w}{\partial t} = A(w)\frac{\partial w}{\partial x} + B(w)\frac{\partial w}{\partial y} + C(y)w, \quad (x, y) \in [0, L] \times [0, D], \quad t \in (0, t_{\text{f}}], \tag{4.44}$$

where $w = (u, v, \phi)^T$ is a vector function, u, v are the velocity components in the x and y directions, respectively, h is the depth of the fluid, g is the acceleration due to gravity, and $\phi = 2\sqrt{gh}$.

The matrices A, B and C have the form

$$A = -\begin{bmatrix} u & 0 & \phi/2 \\ 0 & u & 0 \\ \phi/2 & 0 & u \end{bmatrix}, \quad B = -\begin{bmatrix} v & 0 & 0 \\ 0 & v & \phi/2 \\ 0 & \phi/2 & v \end{bmatrix}, \quad C = \begin{bmatrix} 0 & f & 0 \\ -f & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.45}$$

where f is the Coriolis term

$$f = \hat{f} + \beta(y - D/2), \quad \beta = \frac{\partial f}{\partial y}, \quad \forall\, y \in [0, D], \tag{4.46}$$

with $\hat{f}$ and β constants. We assume periodic solutions in the x direction for all three state variables while in the y direction $v(x, 0, t) = v(x, D, t) = 0, \quad x \in [0, L], \quad t \in (0, t_{\mathrm{f}}]$, and Neumann boundary condition are considered for u and ϕ.

The alternating direction fully implicit scheme was used to construct the high-fidelity forward SWE discrete model. It was implemented in Fortran and uses a sparse matrix environment. For operations with sparse matrices we utilized SPARSEKIT library [151], and the sparse linear systems resulted from quasi-Newton iterations was solved using MGMRES library [152, 153, 154]. This is the model employed for the numerical experiments in [137, See Appendix]. The equations are not decoupled as in [128] where the Jacobian is either block cyclic tridiagonal or block tridiagonal. The discrete tangent linear and adjoint models were derived by hand and their accuracy was verified using the methodology proposed in [155]. Keeping all the equations together allowed us to use the same alternating direction fully implicit scheme to solve the adjoint model. The discrete Jacobian of the model was obtain by discretization first followed by differentiation. Transposing the discrete Jacobian leaded to the discrete formulation of the adjoint model.

4.6.2 Smoothing experimental settings

For this study we selected the computational domain $(L, D) = (6000\text{km}, 4400\text{km})$ with the final time $t_f = 3\text{h}$. The space grid consisted in 31×21 points while for time domain we used 91 time steps. To test the HMC smoothers with SWE model in the context for data assimilation, we utilized 10 observations distributed over the 3 hours data assimilation window. Here the observations are linearly related to model state with $\mathcal{H} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. 4D-Var is carried out in both high-fidelity space (Full 4D-Var)

and reduced-order space (Reduced 4D-Var) against the HMC sampling smoother in the following settings

i) Sampling the high-fidelity space using the original HMC smoother [101] ("Full HMC"),

ii) Sampling the reduced space, i.e. sampling (4.15a) ("Reduced HMC"),

iii) Sampling the high-fidelity space with approximate gradients, i.e. sampling (4.17a) ("Approximate Full HMC"),

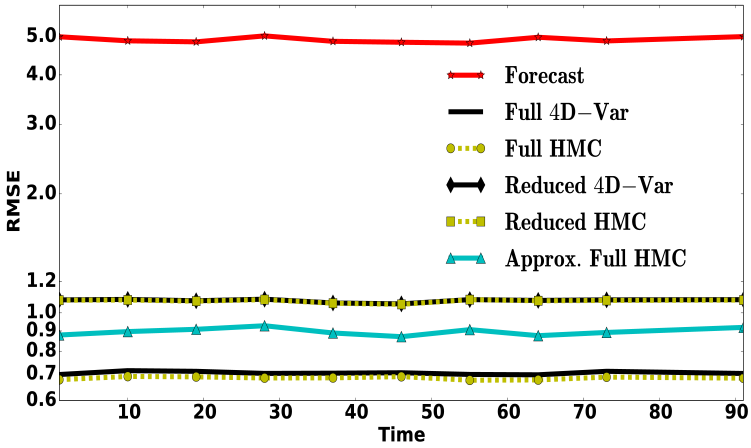
In the three cases, the symplectic integrator used is the position Verlet (4.11) with step size parameters tuned empirically through a preprocessing step. Higher order integrators [70, 62] and automatic tuning of parameters should be considered when theses algorithms are applied to more complicated, e.g., when $\mathcal{H}$ is nonlinear or when the Gaussian prior assumption is relaxed. The reduced basis $\mathbf{V}$ is constructed using initial trajectories of the high-fidelity forward and adjoint models as well as the associated gradient of the full cost function [137] to allow for consistent reduced Karush-Kuhn-Tucker system. Later on this basis is updated using the current proposal and the corresponding trajectories. For each component of the velocity field and geopotential we constructed separated bases. Their sizes were selected to be 45, capturing more than 99% of the spectrum energy of the snapshots matrices. DEIM was implemented inside the POD reduced order models to decrease the computational complexities of the quadratic nonlinearities. For the construction of the reduced order adjoint model we employed the "adjoint of reduced + reduced of adjoint" method with corresponding adjoint test described in [137, See Figure 4a].

4.6.3 Numerical results

Due to the simple settings described above the posterior distribution is not expected to deviate notably from a Gaussian. This will enable us to easily test the quality of the ensemble by testing the first two moments generated from the ensemble.

The mean of the ensemble generated by HMC smoother is an MVUE of the posterior mean, and we are interested in comparing it against the 4D-Var solution. Figure 4.1 shows the Root mean squared (RMSE) errors associated with the 4D-Var and HMC estimates of the posterior mean. The size of the ensemble generated by the different HMC smoothers here is $N_{ens} = 100$. We see clearly that the MVUE generated by HMC in both full and reduced space is at least as good as the 4D-Var minimizer. It is obvious that using Algorithm 6 to sample the full space, while approximating the gradient using reduced-space

Figure 4.1: Data assimilation results using 4D-Var schemes, and HMC smoother, in both high-fidelity space in reduced-order space. Errors for HMC smoother are obtained for 100 ensemble members with 25 burn-in steps, and 5 mixing steps. The steps size for the symplectic integrator is empirically tuned and unified to $T = 0.1$ with $h = 0.01$, and $m = 10$.



information, results in an analysis that is better than the case where the sampler is limited to the reduced space. In addition to testing the quality of the analysis (first-order moment here), we are interested in quantifying the quality of the analysis error covariance matrix generated by HMC. For reference we use HMC in full space to sample $\mathrm{N_{ens}} = 1000$ members to produce a good estimate $\mathbf{A}_0^{\mathrm{ens}} \approx \mathbf{A}_0$. In the cases of reduced space sampling and approximate sampling in the full space we fix the ensemble size to $\mathrm{N_{ens}} = 100$. To compare analysis error covariances obtained in the different scenarios we perform a statistical test of the hypothesis $\mathbf{H}_0 : \Sigma_1 = \Sigma_2$ for the equality of two covariance matrices. Since the state space dimension can be much larger than ensemble size, we choose the test statistic [156] that works in high dimensional settings. Assume we have two probability distributions with covariance matrices Σ_1, Σ_2 respectively, and consider sample estimates $\mathbf{S}_1$, $\mathbf{S}_2$ obtained using ensembles of sizes n_1 and n_2, respectively. The test statistic t^*_{mn} defined in (4.47) asymptotically follows a standard normal distribution in the limit of large ensemble size and state space dimension. At a significance level α, the two sided test

$\mathbf{H}_0 : \Sigma_1 = \Sigma_2$ is rejected only if $|t^*_{mn}| > z_{\alpha/2}$, where $Z = \mathcal{N}(0,)$, and $\mathcal{P}(Z \geq z_{\alpha/2}) = \alpha/2$.

$$
\begin{aligned}
t^*_{mn} &= \frac{t_{mn}}{\hat{\theta}}\,, \\
t_{mn} &= \left(1 - \frac{n_1 - 2}{\eta_1}\mathrm{Tr}\left(\mathbf{S}_1^2\right)\right) + \left(1 - \frac{n_2 - 2}{\eta_2}\mathrm{Tr}\left(\mathbf{S}_2^2\right)\right) - 2\mathrm{Tr}(\mathbf{S}_1\mathbf{S}_2) - \frac{n_1}{\eta_1}\left(\mathrm{Tr}(\mathbf{S}_1)\right)^2 - \frac{n_2}{\eta_2}\left(\mathrm{Tr}(\mathbf{S}_2)\right)^2\,, \\
\eta_1 &= (n_1 + 2)(n_1 - 1), \quad \eta_2 = (n_2 + 2)(n_2 - 1)\,, \\
n &= n_1 + n_2, \quad \mathbf{S} = \frac{n_1}{n}\mathbf{S}_1 + \frac{n_2}{n}\mathbf{S}_2\,, \\
\hat{\theta} &= \sqrt{4a^2\left(\frac{n_1 + n_2}{n_1 n_2}\right)^2}, \quad a = \frac{n^2}{(n+2)(n-1)}\left(\mathrm{Tr}(\mathbf{S}^2) - \frac{(\mathrm{Tr}(\mathbf{S}))^2}{n}\right)\,.
\end{aligned}
\tag{4.47}
$$

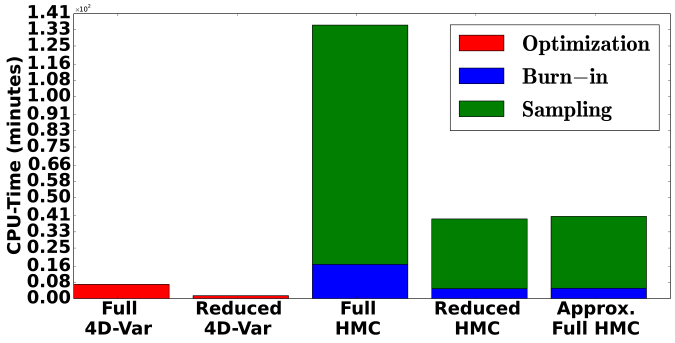
Table 4.1 shows the results of the tests conducted to compare the covariance matrices. In

Table 4.1: Results of statistical tests conducted to compare covariance matrices obtained by HMC smoother in the three scenarios. $\mathbf{A}_0$ is the true posterior covariance of the distribution (4.2). $\widetilde{\mathbf{A}}_0$ is the true posterior covariance of the distribution with negative-log given by (4.15), while $\widetilde{\mathbf{A}}_0^{\mathrm{ens}}$ is the ensemble-based approximation obtained by Algorithm 6. $\widehat{\mathbf{A}}_0$ is the true posterior covariance of the distribution with negative-log given by (4.17), while $\widehat{\mathbf{A}}_0^{\mathrm{ens}}$ is the ensemble-based approximation obtained by Algorithm 6.

Test			Ensemble statistics	Test-statistic
1	Sampling the reduced space	$\mathbf{H_0} : \mathbf{A}_0 = \widetilde{\mathbf{A}}_0$ $\mathbf{H_a} : \mathbf{A}_0 \neq \widetilde{\mathbf{A}}_0$	$n_1 = 1000$, $n_2 = 100$ $\mathbf{S}_1 = \mathbf{A}_0^{\mathrm{ens}}$, $\mathbf{S}_2 = \widetilde{\mathbf{A}}_0^{\mathrm{ens}}$	$t^*_{nm} = 61.0258$
2	Sampling the full space with approximate gradient	$\mathbf{H_0} : \mathbf{A}_0 = \widehat{\mathbf{A}}_0$ $\mathbf{H_a} : \mathbf{A}_0 \neq \widehat{\mathbf{A}}_0$	$n_1 = 1000$, $n_2 = 100$ $\mathbf{S}_1 = \mathbf{A}_0^{\mathrm{ens}}$, $\mathbf{S}_2 = \widehat{\mathbf{A}}_0^{\mathrm{ens}}$	$t^*_{nm} = 2.4514$

the case of sampling in the reduced space the null hypothesis is rejected due to strong evidence based on the samples' estimates. For the approximate full space sampling at a significance level $\alpha = 0.01$ there is no significant evidence to supports rejection. This gives a strong indication that the ensemble generated in the second case describes the uncertainty in the analysis much better than the first case. The test results at least don't oppose the conclusion that sampling (4.17a) using Algorithm 6 results in ensembles capable of estimating the posterior covariance matrix.

Figure 4.2: Data assimilation results using 4D-Var schemes, and HMC smoother, in both high-fidelity space in reduced-order space. CPU-times for HMC smoother are obtained for 30 ensemble members with 25 burn-in steps, and 5 mixing steps. The steps size for the symplectic integrator is empirically tuned and unified to $T = 0.1$ with $h = 0.01$, and $m = 10$. The red color represents the CPU-time spent during optimization steps only. Blue and Green colors, respectively, represent CPU-time spent during the burn-in and the sampling(and mixing) steps.



4.6.4 Computational costs

The computational cost for HMC smoother in full space is much higher than the cost of 4D-Var, however it comes with the advantage of generating a consistent estimate of the analysis error covariance matrix. The bottleneck of HMC smoother is the propagation of the forward and backward model to evaluate the gradient of the potential energy. Using surrogate models radically reduces the computational cost. A detailed discussion of the computational cost in terms of model propagation can be found in [101]. Here we report the CPU time of the different scenarios as shown in Figure 4.2 and Table 4.2. The HMC CPU-time also depends on the settings of the parameters and the size of the ensemble. Following [101] we compare the CPU-times to generate 30 ensemble members. The CPU-times are almost similar when the two strategies in Algorithm 6 are applied, and both are approximately four times faster than the original HMC smoother. The online cost of the approximate smoother is still higher than the cost of 4D-Var in full space, however it is notably reduced by using information coming from a reduced space. The cost can be further reduced by cleverly tuning the sampler parameters or projecting the observation operator and observation error statistics in the reduced space. These ideas will be considered in the future to further reduce the cost of the HMC sampling smoother. It is very important to highlight the fact that the goal is not just to find an analysis state but to approximate the whole posterior distribution. Despite the high cost of the HMC smoother,

we obtain a consistent description of the uncertainty of the analysis state, e.g. an estimate of the posterior covariances.

Table 4.2: Data assimilation results using 4D-Var schemes, and HMC smoother, in both high-fidelity space in reduced-order space. CPU-times for HMC smoother are obtained for 30 ensemble members with 25 burn-in steps, and 5 mixing steps. The steps size for the symplectic integrator is empirically tuned and unified to $T = 0.1$ with $h = 0.01$, and $m = 10$.

Cost	**Experiment**							
	4D-Var		**HMC Smoother**					
	high-fidelity space	**reduced-order space**	**high-fidelity space**		**reduced-fidelity space**		**high-fidelity space with approximate gradient**	
			average per ensemble member	*total*	*average per ensemble member*	*total*	*average per ensemble member*	*total*
CPU-time (minutes)	7.04	1.44	0.68	118.42	0.20	34.50	0.20	35.58

4.7 Conclusions and Future Work

The HMC sampling smoother is developed as a general ensemble-based data assimilation framework to solve the non-Gaussian four-dimensional data assimilation problem. The original formulation of the HMC smoother works with the full dimensional model. It provides a consistent description of the posterior distribution, however it is very expensive due to the necessary large number of full model runs. The reduced order HMC sampling smoother employs reduced-order approximations of the model dynamics. It achieves computational efficiency while retaining most of the accuracy of the full space HMC smoother. The formulations discussed here still assume a Gaussian prior at the initial time, which is a strong assumption since the forward propagation through nonlinear model dynamics will result in a non-Gaussian likelihood. This assumption, however, can be easily relaxed using a mixture of Gaussians to represent the background at the initial time; this will be considered in future work. We plan to explore the possibility of using the KL-Divergence measure between the high fidelity distribution and both the projected and the approximate

posterior distribution, to guide the optimal choice of the size of reduced-order basis. We would also like to explore the possibility of enhancing the quality of the reduced-order ensembles, by incorporating corrections obtained from several reduced-order simulations of different resolutions. In future work we will consider incorporating an HMC sampler capable of automatically tuning the parameters of the symplectic integrator, such as NUTS [74], in order to further enhance the smoother performance.

Chapter 5

Cluster Sampling Filters for Non-Gaussian Data Assimilation

5.1 Introduction

Data assimilation (DA) is a complex process that involves combining information from different sources in order to produce accurate estimates of the true state of a physical system such as the atmosphere. Sources of information include computational models of the system, a background probability distribution, and observations collected at discrete time instances. With model state denoted by $\mathbf{x} \in \mathbb{R}^{N_{state}}$, the prior probability density $\mathbf{P}^b(\mathbf{x})$ encapsulates the knowledge about the system state before incorporating any other source of information such as the observations. Let $\mathbf{y} \in \mathbb{R}^{N_{obs}}$ be a measurement (observation) vector. The observation likelihood function $\mathcal{P}(\mathbf{y}|\mathbf{x})$ quantifies the mismatch between the model predictions (of observed quantities) and the collected measurements. A standard application of Bayes' theorem provides the posterior probability distribution $\mathcal{P}(\mathbf{x}|\mathbf{y})$ that provides an improved description of the unknown true state of the system of interest.

In the ideal case where the underlying probability distributions are Gaussian, the model dynamics is linear, and the observations are linearly related to the model state, the posterior can be obtained analytically for example by applying Kalman filter (KF) equations [53, 54]. For large dimensional problems the computational cost of the standard Kalman filter is prohibitive, and in practice the probability distributions are approximated using small ensembles. The ensemble-based approximation has led to the ensemble Kalman filter (EnKF) family of methods [10, 20, 18, 11]. Several modifications of EnKF, for ex-

ample [19, 21, 9, 17, 22, 8], have been introduced in the literature to solve practical DA problems of different complexities.

One of the drawbacks of the EnKF family is the reliance on an ensemble update formula that comes from the linear Gaussian theory. Several approaches have been proposed in the literature to alleviate the limitations of the Gaussian assumptions. The maximum likelihood ensemble filter (MLEF) [55, 23, 16] computes the maximum a posteriori estimate of the state in the ensemble space. The iterative EnKF [40, 17] (IEnKF) extends MLEF to handle nonlinearity in models as well as in observations. IEnKF, however, assumes that the underlying probability distributions are Gaussian and the analysis state is best estimated by the posterior mode.

These families of filters can generally be tuned (e.g., using inflation and localization) for optimal performance on the problem at hand. However, if the posterior is a multimodal distribution, these filters are expected to diverge, or at best capture a single probability mode, especially in the case of long-term forecasts. Only a small number of filtering methodologies designed to work in the presence of highly non-Gaussian errors are available, and their efficiency with realistic models is yet to be established.

The Hybrid/Hamiltonian Monte Carlo (HMC) sampling filter was proposed in [36] as a fully non-Gaussian filtering algorithm, and has been extended to the four-dimensional (smoothing) setting in [36, 37, 38, 39]. The HMC sampling filter is a sequential DA filtering scheme that works by directly sampling the posterior probability distribution via an HMC approach [48, 80]. The HMC filter is designed to handle cases where the underlying probability distributions are non-Gaussian. Nevertheless, the first HMC formulation presented in [36] assumes that the prior distribution can always be approximated by a Gaussian distribution. This assumption was introduced for simplicity of implementation; however, it can be too restrictive in many cases, and may lead to inaccurate conclusions. This strong assumption needs to be relaxed in order to accurately sample from the true posterior, while preserving computational efficiency.

This work relaxes the Gaussian prior assumption in the original HMC formulation. Specifically, the prior is represented by a Gaussian Mixture Model (GMM) that is fitted to the forecast ensemble via a clustering step. The posterior is formulated accordingly. In the analysis step the resulting mixture posterior is sampled following a HMC approach. The analysis phase, however, can be easily modified to incorporate any other efficient direct sampling method. The resulting algorithm is named the cluster HMC ($C\ell$HMC) sampling filter. In order to improve the sampling from the mixture posterior a more efficient version, named $C\ell$HMC filter (MC-$C\ell$HMC), is also discussed.

Using a GMM to approximate the prior density, given the forecast ensemble, was presented

in [25, 22] as a means to solve the nonlinear filtering problem. In [25], a continuous approximation of the prior density was built as a sum of Gaussian kernels, where the number of kernels is equal to the ensemble size. Assuming a Gaussian likelihood function, the posterior was formulated as a GMM with updated mixture parameters. The updated means and covariance matrices of the GMM posterior were obtained by applying the convolution rule of Gaussians to the prior mixture components and the likelihood, and the analysis ensemble was generated by direct sampling from the GMM posterior. On the other hand, the approach presented in [22] works by fitting a GMM to the prior ensemble with number of mixture components detected using Akaike information criterion. The EnKF equations are applied to each of the components in the mixture distribution to generate an analysis ensemble from the GMM posterior.

Unlike the existing approaches [25, 22], the methodology proposed herein is fully non-Gaussian, and does not limit the posterior density to a Gaussian mixture distribution or Gaussian likelihood functions. Here we sample the posterior distribution using a Hamiltonian Monte Carlo approach, however the direct sampling method can be replaced with any efficient analysis algorithm capable of dealing with high-dimensional multimodal distributions.

The remaining part of the paper is organized as follows. Section 5.2 reviews the original formulation of the HMC sampling filter. Section 5.3 presents the new algorithms $C\ell$HMC and MC-$C\ell$HMC . Numerical results and discussions are presented in Section 5.4. Conclusions are drawn in Section 5.5.

5.2 The HMC Sampling Filter

In this section we present a brief overview of the HMC sampling methodology, followed by the original formulation of the HMC sampling filter.

5.2.1 HMC sampling

HMC sampling follows an auxiliary-variable approach [157, 158] to accelerate the sampling process of Markov chain Monte-Carlo (MCMC) algorithms. In this approach, the MCMC sampler is devoted to sampling the joint probability density of the target variable, along with an auxiliary variable. The auxiliary variable is chosen carefully to allow for the construction of a Markov chain that mixes faster, and is easier to simulate than sampling the marginal density of the target variable [159].

The main component of the HMC sampling scheme is an auxiliary Hamiltonian system that plays the role of the proposal (jumping) distribution. The Hamiltonian dynamical system operates in a phase space of points $(\mathbf{p}, \mathbf{x}) \in \mathbb{R}^{2\mathrm{N_{state}}}$, where the individual variables are the position $\mathbf{x} \in \mathbb{R}^{\mathrm{N_{state}}}$, and the momentum $\mathbf{p} \in \mathbb{R}^{\mathrm{N_{state}}}$. The total energy of the system, given the position and the momentum, is described by the Hamiltonian function $H(\mathbf{p}, \mathbf{x})$. A general formulation of the Hamiltonian function (the Hamiltonian) of the system is given by:

$$H(\mathbf{p}, \mathbf{x}) = \frac{1}{2}\,\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} - \log(\phi(\mathbf{x})) = \frac{1}{2}\,\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} + \mathcal{J}(\mathbf{x})\,, \tag{5.1}$$

where $\mathbf{M} \in \mathbb{R}^{\mathrm{N_{state}}\times\mathrm{N_{state}}}$ is a symmetric positive definite matrix referred to as the *mass matrix*. The first term in the sum (5.1) quantifies the kinetic energy of the Hamiltonian system, while the second term is the associated potential energy.

The dynamics of the Hamiltonian system in time is described by the following ordinary differential equations (ODEs):

$$\frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{p}}\, H\,, \qquad \frac{d\mathbf{p}}{dt} = -\nabla_{\mathbf{x}}\, H. \tag{5.2}$$

The time evolution of the system (5.2) in the phase space is described by the flow: [60, 61]

$$\Phi_T : \mathbb{R}^{2\mathrm{N_{state}}} \to \mathbb{R}^{2\mathrm{N_{state}}}, \quad \Phi_T\big(\mathbf{p}(0), \mathbf{x}(0)\big) = \big(\mathbf{p}(T), \mathbf{x}(T)\big), \tag{5.3}$$

which maps the initial state of the system $(\mathbf{p}(0), \mathbf{x}(0))$ to $(\mathbf{p}(T), \mathbf{x}(T))$, the state of the system at time T. In practical applications, the analytic flow Φ_T is replaced by a numerical solution using a time reversible and symplectic numerical integration method [62, 61]. The length of the Hamiltonian trajectory T can generally be long, and may lead to instability of the time integrator if the step size is set to T. In order to accurately approximate Φ_T, the symplectic integrator typically takes m steps of size $h = T/m$ where h is chosen such as to maintain stability. We will use Φ_T hereafter to represent the numerical approximation of the Hamiltonian flow.

Given the formulation of the Hamiltonian (5.1), the dynamics of the Hamiltonian system is governed by the equations

$$\frac{d\mathbf{x}}{dt} = \mathbf{M}^{-1}\mathbf{p}\,, \qquad \frac{d\mathbf{p}}{dt} = -\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x})\,. \tag{5.4}$$

The canonical probability distribution of the state $(\mathbf{p}, \mathbf{x})$ of the Hamiltonian system in the phase space $\mathbb{R}^{2\mathrm{N_{state}}}$, upto a scaling factor, is given by

$$\exp\left(-H(\mathbf{p}, \mathbf{x})\right) = \exp\left(-\frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p} - \mathcal{J}(\mathbf{x})\right) \propto \exp\left(-\frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}\right)\phi(\mathbf{x}). \tag{5.5}$$

The product form of this joint probability distribution shows that the two variables $\mathbf{p}$, and $\mathbf{x}$ are independent [62]. The marginal distribution of the momentum variable is Gaussian, $\mathbf{p} \sim \mathcal{N}(0, \mathbf{M})$, while the marginal distribution of the position variable is proportional to the negative-logarithm (negative-log) of the potential energy, that is $\mathbf{x} \sim f(\mathbf{x}) \propto \phi(\mathbf{x}) = \exp(-\mathcal{J}(\mathbf{x}))$. Here $f(\mathbf{x})$ is the normalized marginal density of the position variable, while $\phi(\mathbf{x})$ drops the scaling factor (e.g. the normalization constant) of the density function.

In order to draw samples $\{\mathbf{x}(e)\}_{e=1,2,...,N_{ens}}$ from a given probability distribution $f(\mathbf{x}) \propto \phi(\mathbf{x})$, HMC makes the following analogy with the Hamiltonian mechanical system (5.2). The state $\mathbf{x}$ is *viewed* as a position variable, and an auxiliary momentum variable $\mathbf{p} \sim \mathcal{N}(0, \mathbf{M})$ is included. The negative-log of the target probability density $\mathcal{J}(\mathbf{x}) = -\log(\phi(\mathbf{x}))$ is viewed as the potential energy of an auxiliary Hamiltonian system. The kinetic energy of the system is given by the negative-log of the Gaussian distribution of the auxiliary momentum variable. The mass matrix $\mathbf{M}$ is a user-defined parameter that is assumed to be symmetric positive definite. To achieve favorable performance of the HMC sampler, $\mathbf{M}$ is generally assumed to be diagonal, with values on the diagonal chosen to reflect the scale of the components of the target variable under the target density [36, 60]. The HMC sampler proceeds by constructing a Markov chain whose stationary distribution is set to the canonical joint density (5.5). The chain is initialized to some position and momentum values, and at each step of the chain, a Hamiltonian trajectory starting at the current state is constructed to propose a new state. A Metropolis-Hastings-like acceptance rule is used to either accept or reject the proposed state. Since both position and momentum are statistically independent, the retained position samples are actually sampled from the target density $f(\mathbf{x})$. The collected momentum samples are discarded, and the position samples are returned as the samples from the target probability distribution $f(\mathbf{x})$.

The performance of the HMC sampling scheme is greatly influenced by the settings of the Hamiltonian trajectory, that is the choice of the two parameters m, h. The step size h should be small enough to maintain stability, while m should be generally large for the sampler to reach distant points in the state space. The parameters of the Hamiltonian trajectory can be set empirically [60] to achieve an acceptable rejection rate of at most 25% 30% , or be automatically adapted using automatic tuning schemes such as the No-U-Turn sampler(NUTS) [74], or the Riemannian Manifold HMC sampler (RMHMC) [65].

The ideas presented in this work can be easily extended to incorporate any of the HMC sampling algorithms with automatically tuned parameters. In this paper we tune the parameters of the Hamiltonian trajectory following the empirical approach, and focus on the sampler performance due to the choice of the prior distribution in the sequential filtering context.

5.2.2 HMC sampling filter

In the filtering framework, following a perfect-model approach, the posterior distribution $\mathbf{P}^{\mathrm{a}}(\mathbf{x}_k)$ at a time instance t_k follows from Bayes' theorem:

$$\mathbf{P}^{\mathrm{a}}(\mathbf{x}_k) = \mathcal{P}(\mathbf{x}_k|\mathbf{y}_k) = \frac{\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k)\mathbf{P}^{\mathrm{b}}(\mathbf{x}_k)}{\mathcal{P}(\mathbf{y}_k)} \propto \mathcal{P}(\mathbf{y}_k|\mathbf{x}_k)\mathbf{P}^{\mathrm{b}}(\mathbf{x}_k)\,, \tag{5.6}$$

where $\mathbf{P}^{\mathrm{b}}(\mathbf{x}_k)$ is the prior distribution, $\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k)$ is the likelihood function, all at time instance t_k. $\mathcal{P}(\mathbf{y}_k)$ acts as a scaling factor and is ignored in the HMC context.

As mentioned in Section 5.1, the formulation of the HMC sampling filter proposed in [36] assumes that the prior distribution $\mathbf{P}^{\mathrm{b}}(\mathbf{x}_k)$ can be represented by a Gaussian distribution $\mathcal{N}(\mathbf{x}_k^{\mathrm{b}}, \mathbf{B_k})$, that is

$$\mathbf{P}^{\mathrm{b}}(\mathbf{x}_k) = \frac{(2\pi)^{-\frac{\mathrm{N_{state}}}{2}}}{\sqrt{|\mathbf{B}_k|}} \exp\left(-\frac{1}{2}\|\mathbf{x}_k - \mathbf{x}_k^{\mathrm{b}}\|^2_{\mathbf{B}_k^{-1}}\right), \tag{5.7}$$

where $\mathbf{x}_k^{\mathrm{b}}$, is the background state, and $\mathbf{B}_k \in \mathbb{R}^{\mathrm{N_{state}}\times\mathrm{N_{state}}}$ is the background error covariance matrix. The background state $\mathbf{x}_k^{\mathrm{b}}$ is generally taken as the mean of an ensemble of forecasts $\{\mathbf{x}_k^{\mathrm{b}}(e)\}_{e=1,2,...,\mathrm{N_{ens}}}$, obtained by forward model runs from a previous assimilation cycle. The associated weighted norm is defined as:

$$\|\mathbf{a}-\mathbf{b}\|^2_{\mathbf{C}} = (\mathbf{a}-\mathbf{b})^{\mathbf{T}}\mathbf{C}(\mathbf{a}-\mathbf{b}). \tag{5.8}$$

Under the traditional, yet non-restrictive assumption, that the observation errors are distributed according to a Gaussian distribution with zero mean, and observation error covariance matrix $\mathbf{R}_k \in \mathbb{R}^{\mathrm{N_{obs}}\times\mathrm{N_{obs}}}$, the likelihood function takes the form

$$\mathcal{P}(\mathbf{y}_k|\mathbf{x}_k) = \frac{(2\pi)^{-\frac{\mathrm{N_{obs}}}{2}}}{\sqrt{|\mathbf{R}_k|}} \exp\left(-\frac{1}{2}\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}}\right), \tag{5.9}$$

where $\mathcal{H}_k : \mathbb{R}^{\mathrm{N_{state}}} \to \mathbb{R}^{\mathrm{N_{obs}}}$ is the observation operator that maps a given state $\mathbf{x}_k$ to the observation space at time instance t_k. The dimension of the observation space $\mathrm{N_{obs}}$ is generally much smaller than the state space dimension, that is $\mathrm{N_{obs}} \ll \mathrm{N_{state}}$.

The posterior follows immediately from (5.6), (5.7), and (5.9) as:

$$\mathcal{P}^{\mathrm{a}}(\mathbf{x}_k) \propto \phi(\mathbf{x}_k) = \exp\left(-\mathcal{J}(\mathbf{x}_k)\right), \tag{5.10a}$$

$$\mathcal{J}(\mathbf{x}_k) = \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}_k^{\mathrm{b}}\|^2_{\mathbf{B}_k^{-1}} + \frac{1}{2}\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}}\,, \tag{5.10b}$$

where $\mathcal{J}(\mathbf{x}_k)$ is the negative-log of the posterior distribution (5.10a). The derivative of $\mathcal{J}(\mathbf{x}_k)$ with respect to the system state $\mathbf{x}_k$ is given by

$$\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_k) = \mathbf{B}_k^{-1}\left(\mathbf{x}_k - \mathbf{x}_k^{\mathrm{b}}\right) - \mathbf{H}_k^T\,\mathbf{R}_k^{-1}\left(\mathbf{y}_k - \mathcal{H}_k(\mathbf{x})\right), \tag{5.11}$$

where $\mathbf{H}_k = \mathcal{H}_k'(\mathbf{x})$ is the linearized observation operator (e.g. the Jacobian).

The HMC sampling filter [36] proceeds in two steps, namely a forecast step and an analysis step. Given an analysis ensemble of states $\{\mathbf{x}_{k-1}^{\mathrm{a}}(e)\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$ at time t_{k-1}, an ensemble of forecasts at time t_k is generated using the forward model $\mathcal{M}$:

$$\mathbf{x}_k^{\mathrm{b}}(e) = \mathcal{M}_{t_{k-1}\to t_k}\left(\mathbf{x}_{k-1}^{\mathrm{a}}(e)\right), \quad e = 1, 2, \ldots, \mathrm{N_{ens}}. \tag{5.12}$$

In the analysis step, the posterior (5.10) is sampled by running a HMC sampler with potential energy set to (5.10a), where $\mathbf{B}_k$ is approximated using the available ensemble of forecasts.

The formulation of the HMC filter presented in [36], and reviewed above, tends to be restrictive due to the assumption that the prior is always approximated by a Gaussian distribution. The prior distribution can be viewed as the result of propagating the posterior of the previous assimilation cycle using model dynamics. In the case of nonlinear model dynamics, the prior distribution is a nonlinear transformation of a non-Gaussian distribution which is generally expected to be non-Gaussian. Tracking the prior distribution exactly however is not possible, and a relaxation assumption must take place.

We propose conducting a more accurate density estimation of the prior, by fitting a GMM to the available prior ensemble, replacing the Gaussian prior with a Gaussian mixture prior.

5.3 Cluster Sampling Filters

5.3.1 Mixture models

The probability distribution $\mathcal{P}(\mathbf{x})$ is said to be a mixture of $\mathrm{N_c}$ probability distributions $\{C_i(\mathbf{x})\}_{i=1,2,\ldots,\mathrm{N_c}}$, if $\mathcal{P}(\mathbf{x})$ takes the form:

$$\mathcal{P}(\mathbf{x}) = \sum_{i=1}^{\mathrm{N_c}} \tau_i\, C_i(\mathbf{x}) \quad \text{where} \quad \tau_i > 0,\ \forall i \ \text{ and } \ \sum_{i=1}^{\mathrm{N_c}} \tau_i = 1\,. \tag{5.13}$$

The weights τ_i are commonly referred to as the mixing weights, and $C_i(\mathbf{x})$ are the densities of the mixing components.

Gaussian mixture models (GMM)

A GMM is a special case of (5.13) where the mixture components are Gaussian densities, that is $C_i(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \Theta_i)$ with $\Theta_i = \{\mu_i, \Sigma_i\}$ being the parameters of the i^{th} Gaussian component.

Fitting a GMM to a given data set is one of the most popular approaches for density estimation. Given a data set $\{\mathbf{x}(e)\}_{e=1,2,...,\mathrm{N_{ens}}}$, sampled from an unknown probability distribution $\mathcal{P}(\mathbf{x})$, one can estimate the density function $\mathcal{P}(\mathbf{x})$ by a GMM; the parameters of the GMM, i.e. the mixing weights τ_i, the means μ_i, and the covariances Σ_i of the mixture components, can be inferred from the data.

The most popular approach to obtain the maximum likelihood estimate of the GMM parameters is the expectation-maximization (EM) algorithm [160]. EM is an iterative procedure that alternates between two steps, expectation (E) and maximization (M). At iteration $t+1$ the E-step computes the expectation of the complete log-likelihood based on the posterior probability of $\mathbf{x}$ belonging to the i^{th} component, with the parameters $\Theta^{\{t\}}$ from the previous iteration. In particular, the following quantity $Q(\Theta|\Theta^{\{t\}})$ is evaluated:

$$\begin{aligned} Q(\Theta|\Theta^{\{t\}}) &= \sum_{e=1}^{\mathrm{N_{ens}}} \sum_{i=1}^{\mathrm{N_c}} r_{e,i} \, \log\left(\tau_i \, \mathcal{N}\left(\mathbf{x}(e); \Theta_i\right)\right), \\ r_{e,i} &= \frac{\tau_i^{\{t\}} \, \mathcal{N}\left(\mathbf{x}(e); \Theta_i^{\{t\}}\right)}{\sum_{\ell=1}^{\mathrm{N_c}} \tau_\ell^{\{t\}} \, \mathcal{N}\left(\mathbf{x}(e); \Theta_\ell^{\{t\}}\right)}, \\ w_i &= \sum_{e=1}^{\mathrm{N_{ens}}} r_{e,i} \, . \end{aligned} \tag{5.14}$$

Here $\Theta = \{\tau_i, \Theta_i\}_{i=1...\mathrm{N_c}}$ is the parameter set of all the mixture components, and $r_{e,i}$ is the probability that the e^{th} ensemble member lies under the i^{th} mixture component.

In the M-step, the new parameters $\Theta^{\{t+1\}} = \arg\max_\Theta Q$ are obtained by maximizing the conditional probability Q in (5.14) with respect to the parameters Θ. The updated parameters $\Theta^{\{t+1\}}$ are given by the analytical formulas:

$$\begin{aligned} \tau_i^{\{t+1\}} &= \frac{\sum_{e=1}^{\mathrm{N_{ens}}} r_{e,i}}{\mathrm{N_{ens}}} = \frac{w_i}{\mathrm{N_{ens}}}, \\ \mu_i^{\{t+1\}} &= \sum_{e=1}^{\mathrm{N_{ens}}} \mathbf{x}(e) \, \frac{r_{e,i}}{w_i}, \\ \Sigma_i^{\{t+1\}} &= \sum_{e=1}^{\mathrm{N_{ens}}} \left(\mathbf{x}(e) - \mu_i^{\{t+1\}}\right) \left(\mathbf{x}(e) - \mu_i^{\{t+1\}}\right)^T \, \frac{r_{e,i}}{w_i} \, . \end{aligned} \tag{5.15}$$

To initialize the parameters for the EM iterations, the mixing weights are simply chosen to be equal $\tau_i = \mathrm{N_c}^{-1}$, the means μ_i can be randomly selected from the given ensemble, and the covariance matrices of the components can be all set to covariance matrix of the full ensemble. Regardless of the initialization, the convergence of the EM algorithm is ensured by the fact that it monotonically increases the observed data log-likelihood at each iteration [160], that is:

$$\sum_{e=1}^{\mathrm{N_{ens}}} \log\left(\sum_{i=1}^{\mathrm{N_c}} \tau_i^{\{t+1\}} \mathcal{N}\left(\mathbf{x}(e); \Theta_i^{\{t+1\}}\right)\right) \geq \sum_{e=1}^{\mathrm{N_{ens}}} \log\left(\sum_{i=1}^{\mathrm{N_c}} \tau_i^{\{t\}} \mathcal{N}\left(\mathbf{x}(e); \Theta_i^{\{t\}}\right)\right).$$

EM algorithm achieves the improvement of the data log-likelihood indirectly by improving the quantity $Q(\Theta|\Theta^{\{t\}})$ over consecutive iterations, i.e. $Q(\Theta|\Theta^{\{t+1\}}) \geq Q(\Theta|\Theta^{\{t\}})$.

Model selection

Before EM iterations start, the number of mixture components $\mathrm{N_c}$ must be detected. To choose the number of components in the prior mixture model selection is employed. Model selection is a process of selecting a model in the set of a candidate models that gives the best trade-off between model fit and complexity. Here, the best number of components $\mathrm{N_c}$ can be selected with common model selection methodologies such as Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC):

$$\begin{aligned} AIC &= -2\sum_{e=1}^{\mathrm{N_{ens}}} \log\left(\sum_{i=1}^{\mathrm{N_c}} \widehat{\tau}_i \mathcal{N}\left(\mathbf{x}(e); \widehat{\Theta}_i\right)\right) + 2\left(3\mathrm{N_c} - 1\right), \\ BIC &= -2\sum_{e=1}^{\mathrm{N_{ens}}} \log\left(\sum_{i=1}^{\mathrm{N_c}} \widehat{\tau}_i \mathcal{N}\left(\mathbf{x}(e); \widehat{\Theta}_i\right)\right) + \log(\mathrm{N_{ens}})\left(3\mathrm{N_c} - 1\right), \end{aligned} \tag{5.16}$$

where $\{\widehat{\tau}_i, \widehat{\Theta}_i\}_{i=1\ldots\mathrm{N_c}}$ is the set of optimal parameters for the candidate GMM model with $\mathrm{N_c}$ components.

The best number of components $\mathrm{N_c}$ minimizes the AIC or BIC criterion [161, 162]. The main difference between the two criteria, as explained by the second terms in Equation (5.16), is that BIC imposes greater penalty on the number of parameters $(3\mathrm{N_c} - 1)$ of the candidate GMM model. For small or moderate numbers of samples BIC often chooses models that are too simple because of its heavy penalty on complexity.

5.3.2 Cluster HMC sampling filter ($C\ell$HMC)

The prior distribution is approximated by a GMM fitted to the forecast ensemble, e.g., using an EM clustering step. The prior PDF reads:

$$\mathbf{P}^{\mathrm{b}}(\mathbf{x}_k) = \sum_{i=1}^{\mathrm{N_c}} \tau_{k,i}\, \mathcal{N}(\mathbf{x}; \mu_{k,i},\, \Sigma_{k,i}) \;= \sum_{i=1}^{\mathrm{N_c}} \tau_{k,i}\, \frac{(2\pi)^{-\frac{\mathrm{N_{state}}}{2}}}{\sqrt{|\Sigma_{k,i}|}}\, \exp\left(-\frac{1}{2}\|\mathbf{x}-\mu_{k,i}\|^2_{\Sigma_{\mathbf{k,i}}{}^{-1}}\right), \tag{5.17}$$

where the weights $\tau_{k,i}$ quantify the probability that an ensemble member $\mathbf{x}_k(e)$ belongs to the i^{th} component, and $(\mu_{k,i},\, \Sigma_{k,i})$ are the mean and the covariance matrix associated with the i^{th} component of the mixture model at time instance t_k.

Assuming Gaussian observation errors, the posterior can be formulated using equations (5.6), (5.9), and (5.17) as follows:

$$\begin{aligned} f(\mathbf{x}_k) &= \mathbf{P}^{\mathrm{a}}(\mathbf{x}_k) \\ &= \frac{(2\pi)^{-\frac{\mathrm{N_{obs}}}{2}}}{\sqrt{|\mathbf{R}_k|}}\, \exp\left(-\frac{1}{2}\|\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}}\right) \sum_{i=1}^{\mathrm{N_c}} \tau_{k,i}\, \frac{(2\pi)^{-\frac{\mathrm{N_{state}}}{2}}}{\sqrt{|\Sigma_{k,i}|}}\, \exp\left(-\frac{1}{2}\|\mathbf{x}_k-\mu_{k,i}\|^2_{\Sigma_{\mathbf{k,i}}{}^{-1}}\right) \\ &\propto \phi(\mathbf{x}_k) = \sum_{i=1}^{\mathrm{N_c}} \frac{\tau_{k,i}}{\sqrt{|\Sigma_{k,i}|}}\, \exp\left(-\frac{1}{2}\|\mathbf{x}_k-\mu_{k,i}\|^2_{\Sigma_{\mathbf{k,i}}{}^{-1}} - \frac{1}{2}\|\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}}\right). \end{aligned} \tag{5.18}$$

In general the posterior PDF (5.18) will not correspond to a Gaussian mixture due to the nonlinearity of the observation operator. This makes analytical solutions not possible. Here we seek to sample directly from the posterior PDF (5.18).

The HMC sampling requires setting the potential energy term in the Hamiltonian (5.1) to the negative-log of the posterior distribution (5.18). The potential energy term $\mathcal{J}(\mathbf{x}_k)$ is:

$$\begin{aligned} \mathcal{J}(\mathbf{x}_k) &= -\log\left(\sum_{i=1}^{\mathrm{N_c}} \frac{\tau_{k,i}}{\sqrt{|\Sigma_{k,i}|}}\, \exp\left(-\frac{1}{2}\|\mathbf{x}_k-\mu_{k,i}\|^2_{\Sigma_{\mathbf{k,i}}{}^{-1}} - \frac{1}{2}(\|\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}}\right)\right) \\ &= \frac{1}{2}\|\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}} - \log\left(\sum_{i=1}^{\mathrm{N_c}} \frac{\tau_{k,i}}{\sqrt{|\Sigma_{k,i}|}}\, \exp\left(-\frac{1}{2}\|\mathbf{x}_k-\mu_{k,i}\|^2_{\Sigma_{\mathbf{k,i}}{}^{-1}}\right)\right) \qquad (5.19a) \\ &= \frac{1}{2}\|\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}} - \log\left(\sum_{i=1}^{\mathrm{N_c}} \frac{\tau_{k,i}}{\sqrt{|\Sigma_{k,i}|}}\, \exp\left(-\mathcal{J}_{k,i}(\mathbf{x}_k)\right)\right), \end{aligned}$$

where

$$\mathcal{J}_{k,i}(\mathbf{x}_k) = \frac{1}{2}\|\mathbf{x}_k-\mu_{k,i}\|^2_{\Sigma_{\mathbf{k,i}}{}^{-1}}. \tag{5.19b}$$

Equation (5.19) is expected to suffer from numerical difficulties due to evaluating the logarithm of a sum of very small values. To address the accumulation of roundoff errors, and without loss of generality, we assume from now on that the terms in Equation (5.19) under the sum are sorted in decreasing order, i.e.

$$(\tau_{k,i}/\sqrt{|\Sigma_{k,i}|})\exp\left(-\mathcal{J}_{k,i}(\mathbf{x}_k)\right) > (\tau_{k,i+1}/\sqrt{|\Sigma_{k,i+1}|})\exp\left(-\mathcal{J}_{k,i+1}(\mathbf{x}_k)\right), \forall\, i = 1,\ldots,\mathrm{N_c}-1$$

The potential energy function (5.19) is rewritten as:

$$\begin{aligned}\mathcal{J}(\mathbf{x}_k) &= \frac{1}{2}\|\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}} \qquad (5.20a)\\ &\quad - \left[\log\left(\frac{\tau_{k,1}\,\exp\left(-\mathcal{J}_{k,1}(\mathbf{x}_k)\right)}{\sqrt{|\Sigma_{k,1}|}}\right) + \log\left(1+\sum_{i=2}^{\mathrm{N_c}}\frac{\frac{\tau_{k,i}}{\sqrt{|\Sigma_{k,i}|}}\exp\left(-\mathcal{J}_{k,i}(\mathbf{x}_k)\right)}{\frac{\tau_{k,1}}{\sqrt{|\Sigma_{k,1}|}}\exp\left(-\mathcal{J}_{k,1}(\mathbf{x}_k)\right)}\right)\right]\\ &= \frac{1}{2}\|\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}} + \mathcal{J}_{k,1}(\mathbf{x}_k) \qquad (5.20b)\\ &\quad - \log\left(\frac{\tau_{k,1}}{\sqrt{|\Sigma_{k,1}|}}\right) - \log\left(1+\sum_{i=2}^{\mathrm{N_c}}\frac{\tau_{k,i}\,\sqrt{|\Sigma_{k,1}|}}{\tau_{k,1}\,\sqrt{|\Sigma_{k,i}|}}\exp\left(\mathcal{J}_{k,1}(\mathbf{x}_k)-\mathcal{J}_{k,i}(\mathbf{x}_k)\right)\right).\end{aligned}$$

The gradient of the potential energy (5.20) is:

$$\begin{aligned}\nabla_{\mathbf{x}_k}\mathcal{J}(\mathbf{x}_k) &= \mathbf{H}_k^T\mathbf{R}_k^{-1}(\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k) + \nabla_{\mathbf{x}_k}\mathcal{J}_{k,1}(\mathbf{x}_k) \qquad (5.21a)\\ &\quad - \frac{1}{\left(1+\sum_{i=2}^{\mathrm{N_c}}\frac{\tau_{k,i}\,\sqrt{|\Sigma_{k,1}|}}{\tau_{k,1}\,\sqrt{|\Sigma_{k,i}|}}\exp\left(\mathcal{J}_{k,1}(\mathbf{x}_k)-\mathcal{J}_{k,i}(\mathbf{x}_k)\right)\right)}\\ &\quad \cdot\sum_{i=2}^{\mathrm{N_c}}\left\{\frac{\tau_{k,i}\,\sqrt{|\Sigma_{k,1}|}}{\tau_{k,1}\,\sqrt{|\Sigma_{k,i}|}}\exp\left(\mathcal{J}_{k,1}(\mathbf{x}_k)-\mathcal{J}_{k,i}(\mathbf{x}_k)\right)\cdot\right.\\ &\quad \left.\cdot\left[\nabla_{\mathbf{x}_k}\mathcal{J}_{k,1}(\mathbf{x}_k)-\nabla_{\mathbf{x}_k}\mathcal{J}_{k,i}(\mathbf{x}_k)\right]\right\},\end{aligned}$$

where $\nabla_{\mathbf{x}_k}\mathcal{J}_{k,i}(\mathbf{x}_k)$ is given by:

$$\nabla_{\mathbf{x}_k}\mathcal{J}_{k,i}(\mathbf{x}_k) = \Sigma_{k,i}^{-1}(\mathbf{x}_k-\mu_{k,i}) \qquad \forall\, i = 1,2,\ldots,\mathrm{N_c}. \qquad (5.21b)$$

In the case where the mixture contains a single component (one Gaussian distribution) the potential energy function (5.20) and it's gradient (5.21) reduce to the following, respectively:

$$\begin{aligned}\mathcal{J}(\mathbf{x}_k) &= \frac{1}{2}\|\mathbf{x}_k-\mathbf{x}_k^{\mathrm{b}}\|^2_{\mathbf{B}_k^{-1}} + \frac{1}{2}\|\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\|^2_{\mathbf{R}_k^{-1}},\\ \nabla_{\mathbf{x}_k}\mathcal{J}(\mathbf{x}_k) &= \mathbf{B}_k^{-1}\left(\mathbf{x}_k-\mathbf{x}_k^{\mathrm{b}}\right) + \mathbf{H}_k^T\mathbf{R}_k^{-1}\left(\mathcal{H}_k(\mathbf{x}_k)-\mathbf{y}_k\right).\end{aligned} \qquad (5.22)$$

This shows that the $C\ell$HMC sampling filter proposed herein reduces to the original HMC filter the EM algorithm detects a single component during the prior density approximation phase.

The $C\ell$HMC sampling algorithm. As in the HMC sampling filter, information about the analysis probability density at the previous time t_{k-1} is captured by the analysis ensemble of states $\{\mathbf{x}^{\mathrm{a}}_{k-1}(e)\}_{e=1,\ldots,\mathrm{N_{ens}}}$. The forecast step consists of two stages. First, the model (5.12) is used to integrate each analysis ensemble member forward to time t_k where observations are available. Next, a clustering scheme (e.g., EM) is used to generate the parameters of the GMM. The analysis step constructs a Markov chain starting from an initial state $\mathbf{x}^0_k$, and proceeds by sampling the posterior PDF (5.18) at stationarity. Here the superscript over $\mathbf{x}_k$ refers to the iteration number in the Markov chain.

The steps of the $C\ell$HMC sampling filter are detailed in Algorithm 7. As discussed in [36], Algorithm 7 can be used either as a non-Gaussian filter, or as a replenishment tool for parallel implementations of the traditional filters such as EnKF.

5.3.3 Computational considerations

To initialize the Markov chain one seeks a state that is likely with respect to the analysis distribution. Therefore one can start with the background ensemble mean, or with the mean of the component that has the highest weight. Alternatively, one can apply a traditional EnKF step and use the mean analysis to initialize the chain.

The joint ensemble mean and covariance matrix can be evaluated using the forecast ensemble, or using the GMM parameters. Given the GMM parameters $(\tau_{k,i};\, \mu_{k,i},\, \Sigma_{k,i})$, the joint background mean and covariance matrix are, respectively:

$$\overline{\mathbf{x}}^{\mathrm{b}}_k = \sum_{i=1}^{\mathrm{N_c}} \tau_{k,i}\, \mu_{k,i}\,, \tag{5.24a}$$

$$\mathbf{B}^{\mathrm{ens}}_k = \sum_{i=1}^{\mathrm{N_c}} \tau_{k,i}\, \Sigma_{k,i} + \sum_{i=1}^{\mathrm{N_c}} \tau_{k,i}\, (\mu_{k,i} - \overline{\mathbf{x}}^{\mathrm{b}}_k)(\mu_{k,i} - \overline{\mathbf{x}}^{\mathrm{b}}_k)^T\,. \tag{5.24b}$$

Both the potential energy (5.20) and it's gradient (5.21) require evaluating the determinants of the covariance matrices associated with the mixture components. This is a computationally expensive process that is best avoided for large-scale problems. A simple remedy is to force the covariance matrices $\Sigma_{k,i}$, $\forall i = 1, 2, \ldots, \mathrm{N_c}$ to be diagonal while constructing the GMM.

Algorithm 7 Cluster HMC sampling filter ($C\ell$HMC)

1: **Forecast step:** given an analysis ensemble $\{\mathbf{x}^{\mathrm{a}}_{k-1}(e)\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$ at time t_{k-1};

i- generate the forecast ensemble using the model $\mathcal{M}$:

$$\mathbf{x}^{\mathrm{b}}_{k}(e) = \mathcal{M}_{t_{k-1}\rightarrow t_k}\left(\mathbf{x}^{\mathrm{a}}_{k-1}(e)\right), \quad e = 1, 2, \ldots, \mathrm{N_{ens}}.$$

ii- Use AIC/BIC criteria to detect the number of mixture components $\mathrm{N_c}$ in the GMM, then use EM to estimate the GMM parameters $\{(\tau_{k,i};\, \mu_{k,i},\, \Sigma_{k,i})\}_{i=1,2,\ldots,\mathrm{N_c}}$.

2: **Analysis step:** given the observation $\mathbf{y}_k$ at time point t_k , follow the steps *i* to *v*:

i- Initialize the Markov Chain ($\mathbf{x}^0_k$) to be to the best estimate available, e.g. to the mean of the joint forecast ensemble, or the mixture component mean with maximum likelihood.

ii- Choose a positive definite mass matrix $\mathbf{M}$.

iii- Set the potential energy function to (5.20), and it's derivative to (5.21).

iv- Initialize the chain with a state $\mathbf{x}^0_k$ and generate $\mathrm{N_{ens}}$ ensemble members from the posterior distribution (5.18) as follows:

1) Draw a random vector $\mathbf{p}^r \sim \mathcal{N}(0, \mathbf{M})$.

2) Use a symplectic numerical integrator (e.g. Verlet, 2-Stage, or 3-Stage [62, 36]) to advance the current state $(\mathbf{p}^r,\, \mathbf{x}^r_k)$ by a pseudo-time increment T to obtain a *proposal* state $(\mathbf{p}^*,\, \mathbf{x}^*_k)$:

$$(\mathbf{p}^*,\, \mathbf{x}^*_k) = \Phi_T\big((\mathbf{p}^r,\, \mathbf{x}^r_k)\big). \tag{5.23}$$

3) Evaluate the energy loss : $\Delta H = H(\mathbf{p}^*,\, \mathbf{x}^*_k) - H(\mathbf{p}^r,\, \mathbf{x}^r_k)$.

4) Calculate the acceptance probability: $a^{(r)} = 1 \wedge e^{-\Delta H}$.

5) Discard both $\mathbf{p}^*,\, \mathbf{p}^r$.

6) **(Acceptance/Rejection)** Draw a uniform random variable $u^{(r)} \sim \mathcal{U}(0, 1)$:

i- If $a^{(r)} > u^{(r)}$ accept the proposal as the next sample: $\mathbf{x}^{r+1}_k := \mathbf{x}^*_k$;

ii- If $a^{(r)} \leq u^{(r)}$ reject the proposal and continue with the current state: $\mathbf{x}^{r+1}_k := \mathbf{x}^r_k$.

7) Repeat steps 1 to 6 until $\mathrm{N_{ens}}$ distinct samples are drawn.

v- Use the generated samples $\{\mathbf{x}^{\mathrm{a}}_k(e)\}_{e=1,2,\ldots,\mathrm{N_{ens}}}$ as an analysis ensemble.

3: Increase time $k := k + 1$ and repeat steps 1 and 2.

When the Algorithm 7 is applied sequentially at some steps a single mixture component could be detected in the prior ensemble. In this case forcing a diagonal covariance structure does not help; in this case the ensemble covariance is calculated and the standard HMC sampler step is applied.

5.3.4 A multi-chain version of the $C\ell$HMC filter (MC-$C\ell$HMC)

Given the special geometry of the posterior mixture distribution one can construct separate Markov chains for different components of the posterior. These chains can run in parallel to independently sample different regions of the analysis distribution. By running a Markov chain starting at each component of the mixture distribution we ensure that the proposed algorithm navigates all modes of the posterior, and covers all regions of high probability.

The parameters of the jumping distribution for each of the chains can be tuned locally based on the statistics of the ensemble points belonging to the corresponding component in the mixture. This approach is potentially very efficient, not only because it reduces the total running time of the sampler, but also because it favors an increase acceptance rate. The local ensemble size (sample size per chain) can be specified based on the prior weight of the corresponding component multiplied by the likelihood of the mean of that component. Every chain is initialized to the mean of the corresponding component in the prior mixture. The diagonal of the mass matrix can be set globally for all components, for example using the diagonal of the precision matrix of the forecast ensemble, or can be chosen locally based on the second-order moments estimated from the prior ensemble under the corresponding component in the prior mixture. This local choice of the mass matrix does not change the marginal density of the target variable.

5.4 Numerical Results

We first apply the proposed algorithms, $C\ell$HMC and MC-$C\ell$HMC to sample a simple one-dimensional mixture distribution. The proposed methodologies are then tested using a quasi-geostrophic (QG) model and compared against the original HMC sampling filter and against EnKF. We mainly use a nonlinear 1.5-layer reduced-gravity QG model with double-gyre wind forcing and bi-harmonic friction [163].

5.4.1 One-dimensional test problem

We start with a prior ensemble generated from a GMM with $\mathrm{N_c} = 5$ and the following mixture parameters:

$$\{(\tau_i;\, \mu_i, \sigma_i^2)\}_{i=1,\dots,5} = \{(0.2;\, -2.4, 0.05),\ (0.1;\, -1.0, 0.07),\ (0.1;\, 0, 0.02),\\ (0.3;\, 1.0, 0.06),\ (0.3;\, 2.4, 0.1)\}\,. \tag{5.25}$$

The EM algorithm is used to construct a GMM approximation of the true probability distribution from which the given prior ensemble is drawn. The model selection criterion used here is AIC. The generated GMM approximation of the prior has $\mathrm{N_c} = 4$ and the following parameters:

$$\{(\tau_i;\, \mu_i, \sigma_i^2)\}_{i=1,\dots,4} = \{(0.169;\, -2.370, 0.052),\ (0.278;\, -0.727, 0.423),\\ (0.229;\, 1.070, 0.065),\ (0.324;\, 2.436, 0.159)\}\,. \tag{5.26}$$

The prior ensemble and the GMM approximation of the true prior are shown in Figure 5.1. Assuming the observation likelihood function is given by:

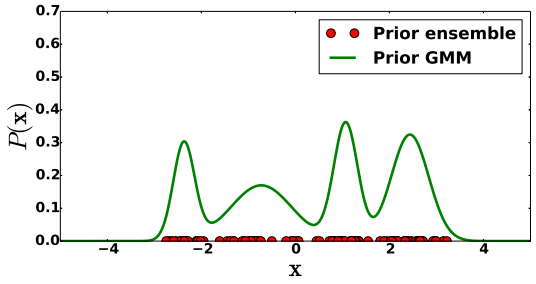
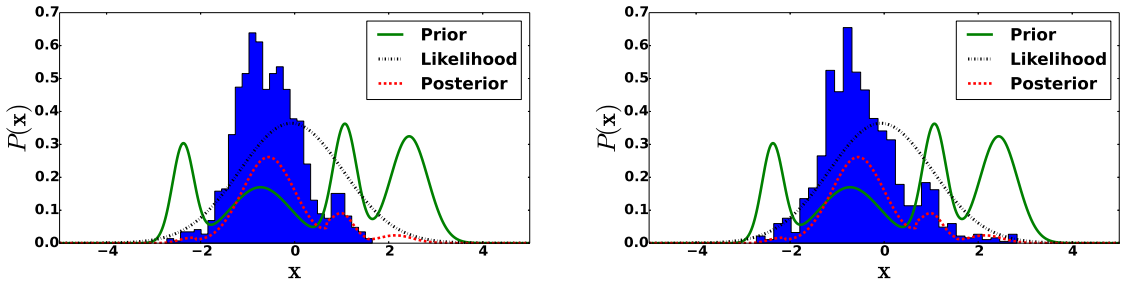


Figure 5.1: The one-dimensional example. A random sample of size $\mathrm{N_{ens}} = 100$ generated from a GMM with parameters given by (5.25), and a GMM constructed by EM algorithm with AIC model selection criterion.

$$\mathcal{P}(\mathbf{y}|\mathbf{x}) = \frac{1}{\sqrt{1.2}\,\sqrt{2\pi}}\ \exp\left(-\frac{1}{2}\,\frac{(\mathbf{x}-\mathbf{y})^2}{1.2}\right), \tag{5.27}$$

with an observation $\mathbf{y} = -0.06858$, the posterior and the histograms of 1000 sample points generated by $C\ell$HMC , and MC-$C\ell$HMC algorithms, are shown in Figure 5.2. In this example, the symplectic integrator used is Verlet with pseudo-time stepping parameters $T = mh$ with $m = 20$, and $h = 0.05$. Since the chains are initialized to the means of the prior mixture components, the burn-in stage is waived, i.e.. the number of burn-in steps

is set to zero. To reduce the correlation between the ensemble members of one chain we discard 15 states (mixing steps) between each two consecutive sampled points. In the MC-$C\ell$HMC filter, the ensemble size per component (per chain) is set to $\mathrm{N}_{\mathrm{ens}} \times \ell_i \times \tau_i$, where ℓ_i is the likelihood of the mean of the i^{th} component in the prior mixture. The results reported



(a) $C\ell$HMC (b) MC-$C\ell$HMC

Figure 5.2: The one-dimensional example. A GMM, a Gaussian likelihood, and the resulting posterior, along with histograms of 1000 sample points generated by the $C\ell$HMC , and the MC-$C\ell$HMC sampling algorithms. The sampling schemes are indicated under each panel. The symplectic integrator used is Verlet with pseudo-time stepping parameters $T = mh$ with $m = 20$, and $h = 0.045$. The number of burn-in steps is zero, and the number of mixing steps is 15.

in Figures 5.2(a) and 5.2(b) show that both $C\ell$HMC and MC-$C\ell$HMC algorithms are capable of generating ensembles with mass distribution accurately representing the underlying target posterior. $C\ell$HMC however fails to sample one of the probability modes, while MC-$C\ell$HMC generates samples from the vicinities of all posterior probability modes.

An implementation of the $C\ell$HMC and MC-$C\ell$HMC sampling algorithms is available from [164].

5.4.2 Quasi-geostrophic model

We employ the QG-1.5 model described by Sakov and Oke [163]. This model is a numerical approximation of the equations:

$$\begin{aligned} q_t &= \psi_x - \varepsilon J(\psi, q) - A\Delta^3\psi + 2\pi \sin(2\pi y)\,, \\ q &= \Delta\psi - F\psi\,, \\ J(\psi, q) &\equiv \psi_x q_x - \psi_y q_y\,, \end{aligned} \tag{5.28}$$

where $\Delta := \partial^2/\partial x^2 + \partial^2/\partial y^2$ and ψ is either the stream function or the surface elevation. We use the values of the model coefficients (5.28) from [163], as follows: $F = 1600$, $\varepsilon = 10^{-5}$, and $A = 2 \times 10^{-12}$. The domain of the model is a 1×1 [space units] square, with $0 \leq x \leq 1$, $0 \leq y \leq 1$, and is discretized by a grid of size 129×129 (including boundaries). Boundary conditions used are $\psi = \Delta\psi = \Delta^2\psi = 0$. The model state dimension is $\mathrm{N}_{\mathrm{state}} = 16641$, while the model trajectories belong to affine subspaces with dimensions of the order of $10^2 - 10^3$ [163].

The time integration scheme used is the fourth-order Runge-Kutta scheme with a time step 1.25 [time units].

For all experiments in this work, the model is run over 1000 model time steps, with observations made available every 10 time steps. In this synthetic model the scales are not relevant, and we use generic space, time, and solution amplitude units.

Observations and observation operators

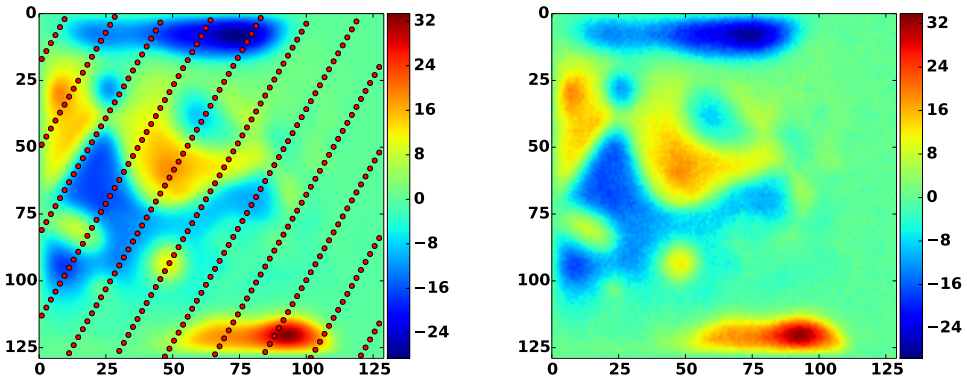
Two observation operators are used with this model.

- First we use a standard linear operator to observe 300 components of ψ. The observation error variance is 4.0 [units squared]. Synthetic the observations are obtained by adding white noise to measurements of the see height level (SSH) extracted from a model run with lower viscosity.
- The second observation operator measures the magnitude of the flow velocity $\sqrt{u^2 + v^2}$. The flow velocity components u, v are obtained using a finite difference approximation of the following relations to the stream function:

$$u = +\frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}. \tag{5.29}$$

In both cases, the observed components are uniformly distributed over the state vector length, with a random offset, that is updated at each assimilation cycle.

The reference initial state along with an example of the observational grid used, and the initial forecast state are shown in Figure 5.3.



(a) Reference initial state and observational grid

(b) Mean of the initial forecast ensemble

Figure 5.3: The QG-1.5 model. The red dots in panel 5.3(a) indicate the location of observations for one of the test cases employed.

Filter tuning

We used a deterministic implementation of EnKF (DEnKF) with parameters tuned as suggested in [163]. Specifically, we apply a covariance localization by means of a Hadamard product as explained in [21]. The localization function used is Gaspari-Cohn [165] with localization radius set to 12 grid cells. Inflation is applied with factor $\delta = 1.06$ to the analysis ensemble of anomalies at the end of each assimilation cycle of DEnKF.

The parameters of the HMC and $C\ell$HMC sampling filters are tuned empirically in a preprocessing step in the HMC filter to guarantee a rejection rate at most between 25% to 30%. Here we tune the parameters of the Hamiltonian trajectory only once at the beginning of the assimilation experiment. Specifically, the step size parameters of the symplectic integrator are set to $h = 0.075$, $m = 25$ in the presence of the linear observation operator, and are set to $h = 0.015$, $m = 25$ when the nonlinear observation operator (5.29) is used. The integrator used for the Hamiltonian system in all experiments is the three-stage symplectic integrator [36, 62]. The mass matrix $\mathbf{M}$ is chosen to be a diagonal matrix whose diagonal is set to the diagonal of the precision matrix of the forecast ensemble. In the current experiments, the first 50 steps of the Markov chains are discarded as a burn-in stage. Alternatively, one can run a suboptimal minimization of the negative-log of the posterior to achieve convergence to the posterior.

The parameters of the MC-$C\ell$HMC filter are set as follows. The step size parameters of

the symplectic integrator are set to $h = 0.05/\mathrm{N_c}$, $m = 15$ in the experiments with linear observation operator, and $h = 0.0075/\mathrm{N_c}$, $m = 15$ in the case of the nonlinear observation operator (5.29). The mass matrix is a diagonal matrix whose diagonal is set to the diagonal of the precision matrix of the forecast ensemble labeled under the corresponding mixture component. To avoid numerical problems related to very small ensemble variances, for example in the case of outliers, the variances are averaged with the modeled forecast variances of 5 [units squared].

The prior GMM is built with number of components determined using AIC model selection criteria, with a lower bound of 5 of the number of ensemble members belonging to each component of the mixture. This lower bound is enforced as a means to ameliorate the effect of outliers on the GMM construction. In all experiments involving $C\ell$HMC , and MC-$C\ell$HMC , the diagonal covariances relaxation assumption is imposed. However, this structure is not imposed if only one mixture component is detected, and $C\ell$HMC and MC-$C\ell$HMC filters fall back to the original HMC filter. For cases where a component contains a very small number of ensemble members covariance tapering [166] can prove useful.

Assessment metrics

To assess the accuracy of the tested filters we use the root mean squared error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N_{state}}} \sum_{i=1}^{\mathrm{N_{state}}} (x_i - x_i^{\mathrm{true}})^2}\,, \tag{5.30}$$

where $\mathbf{x}^{\mathrm{true}} = \psi^{\mathrm{true}}$ is the reference state of the system and $\mathbf{x}$ is the analysis state, e.g. the average of the analysis ensemble. Here $\mathrm{N_{state}} = 129 \times 129 = 16641$ is the dimension of the model state.We also use Talagrand (rank) histogram [72, 167] to assess the quality of the ensemble spread around the true state.

Results with linear observation operator

Figure 5.4 presents the RMSE (5.30) results of the analyses obtained using EnKF, HMC, $C\ell$HMC , and MC-$C\ell$HMC filters in the presence of a linear observation operator. Figure 5.4 shows that the results of all HMC filter versions improve quickly at the first few assimilation windows. While the results of the original HMC filter improve quickly at the first few assimilation windows, the performance of the original HMC filter degrades

compared to the DEnKF filter performance especially in the long run. We believe that the two main factors contribute to the HMC filter degradation are the parameter tuning, and the development of non-Gaussianity in the prior distribution. The $C\ell$HMC analysis drifts away quickly from the true trajectory. This is mainly because the HMC sampling strategy is unable to cover all probability modes in the posterior distribution. To guarantee that the sampling filter covers the truth well, the sampler has to be able to sample properly from all posterior probability modes. This is achieved, by the design, by the MC-$C\ell$HMC filter. The MC-$C\ell$HMC version produces RMSE results comparable to the RMSE obtained by DEnKF.

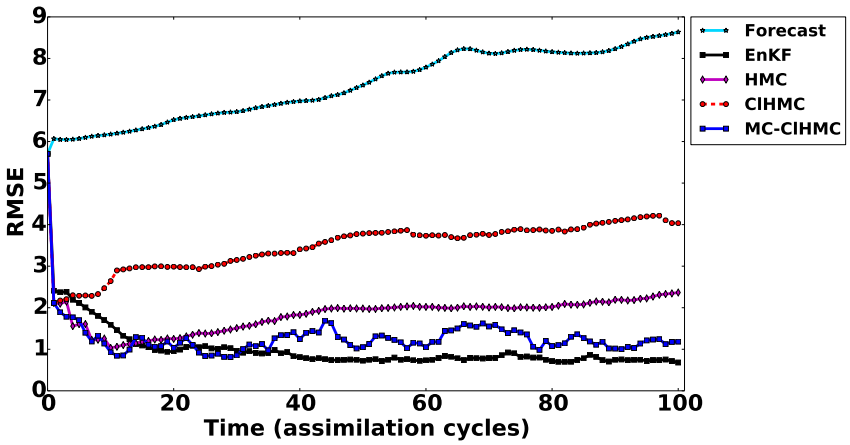


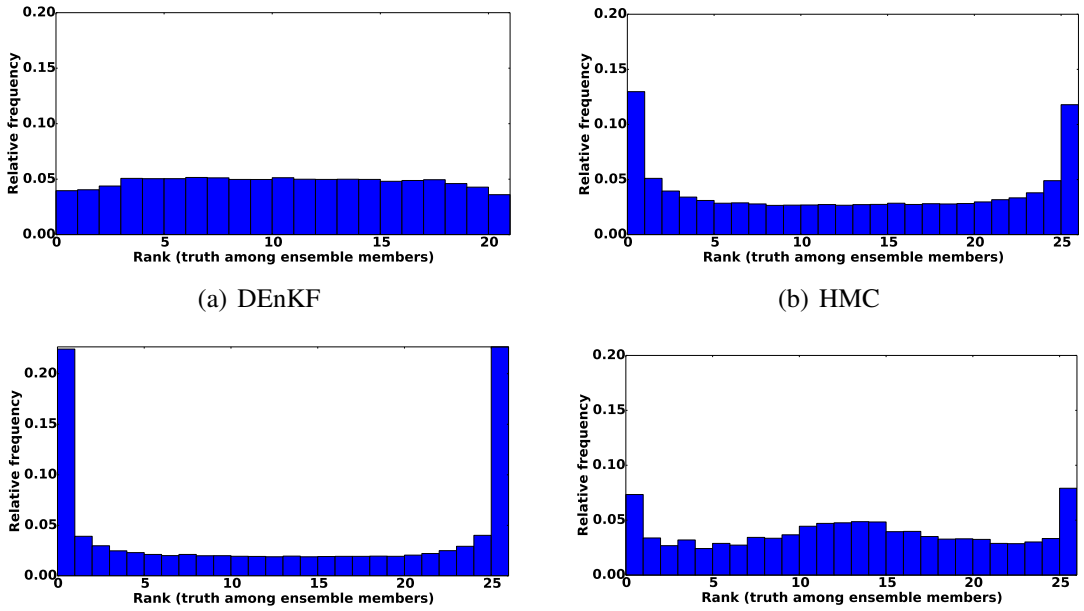
Figure 5.4: Data assimilation results with the linear observation operator. RMSE of the (5.30) analyses obtained by EnKF, HMC, $C\ell$HMC , and MC-$C\ell$HMC filters.

As discussed in [36] the performance of HMC filter can be further enhanced by automatically tuning the parameters of the symplectic integrator at the beginning of each assimilation cycle. Here however we are mainly interested in assessing the performance of the new methodologies compared to the original HMC filter using equivalent settings.

It is important to note that the MC-$C\ell$HMC filter requires shorter Hamiltonian trajectories to explore the space under each local mixture component, which results in computational savings. Additional savings can be obtains by running the chains in parallel to sample different regions of the posterior.

Since we are not interested in only a *single* best estimate of the true state of the system, RMSE alone is not sufficient to judge the quality of the filtering system. The analysis ensemble sampled from the posterior should be spread widely enough to cover the truth and avoid filter collapse. The rank histograms of the analysis ensembles are shown in Fig-

ure 5.5. The two small spikes in Figure 5.5(b) suggest that the performance of the original HMC filter could be enhanced by increasing the length of the Hamiltonian trajectories in some assimilation cycles.



(a) DEnKF (b) HMC

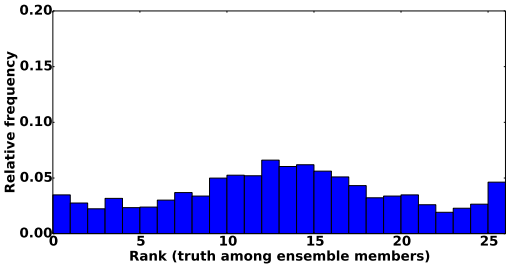
(c) $C\ell$HMC +AIC (d) MC-$C\ell$HMC +AIC

Figure 5.5: Data assimilation results with the linear observation operator. The rank histograms of where the truth ranks among posterior ensemble members. The ranks are evaluated for every 16^{th} variable in the state vector (past the correlation bound) at 100 assimilation times.

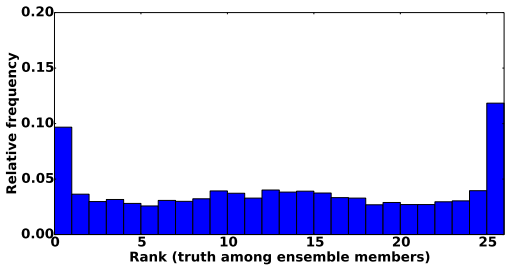
The rank histogram shown in Figure 5.5(c) shows that the analysis ensembles produced by the $C\ell$HMC filter tend to be under-dispersed. Since the ensemble size is relatively small and the prior GMM is multimodal, with regions of low-probability between the different mixture components, a multimodal mixture posterior with isolated components is obtained. As explained in [168], this is a case where HMC sampling in general can suffer from being entrapped in a local minimum (and fails to jump between different high probability modes). This behavior is expected to result in ensemble collapse, as seen in Figure 5.5(c), leading to filter degradation in the long run as illustrated by the RMS errors shown in Figure 5.4.

The results shown in Figure 5.5(c) suggest that the analysis ensemble collected by $C\ell$HMC

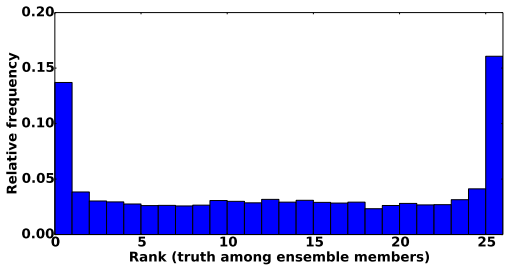
fails to cover all mixture components, thereby losing its dispersion when it is applied repetitively. This is supported by the results in Figure 5.6, where the rank histograms are plotted using results from the first two, five, and ten cycles, respectively.



(a) $C\ell$HMC +AIC; first two observation windows



(b) $C\ell$HMC +AIC; first five observation windows



(c) $C\ell$HMC +AIC; first ten observation windows

Figure 5.6: Data assimilation results using a linear observation operator. Rank histograms of where the truth ranks among posterior ensemble members. The ranks are evaluated for every 16^{th} variable in the state vector (past the correlation bound). Rank histograms of $C\ell$HMC results obtained at the first two, five, and ten assimilation cycles respectively are shown. The model selection criterion used is AIC.

The ensemble collapse can be avoided if we force the sampler to collect ensemble members from all the probability modes. This is illustrated by the rank histograms of results obtained using the MC-$C\ell$HMC filter with AIC criteria as shown in Figure 5.5(d).

We believe that having isolated regions of high probability, e.g., with very small number of ensemble members in each component, can be the critical factor leading the poor long-term performance of $C\ell$HMC . This is alleviated here by imposing a minimum number of 3 ensemble points in each component, e.g. via hard assignment, of the mixture while constructing the GMM approximation of the prior.

With automatic tuning of the Hamiltonian parameters, the performance of both HMC, and

MC-$C\ell$HMC filters is expected to be greatly enhanced.

While we have shown the results of $C\ell$HMC , and MC-$C\ell$HMC with AIC information criterion, experiments carried out using other model selection criteria such as BIC have proven to be very similar.

To help decide whether to apply the original formulation of the HMC filter, or the proposed methodology, one can run tests of non-Gaussianity on the forecast ensemble. To asses non-Gaussianity of the forecast several numeric or visualization normality tests are available, e.g., the Mardia test [169] based on multivariate extensions of skewness and kurtosis measures. Indication of non-Gaussianity can be found by visually inspecting several bivariate contour plots of the joint distribution of selected components in the state vector. Visualization methods for multivariate normality assessment such as chi-square QQ-plots can be very useful as well. Figure 5.7 shows several chi-square QQ-plots of the forecast ensembles generated from the result of EnKF, HMC, and MC-$C\ell$HMC filters at different time instances. These plots show strong signs of non-Gaussianity in the forecast ensemble, and suggest that the Gaussian-prior assumption may in general lead to inaccurate conclusions.

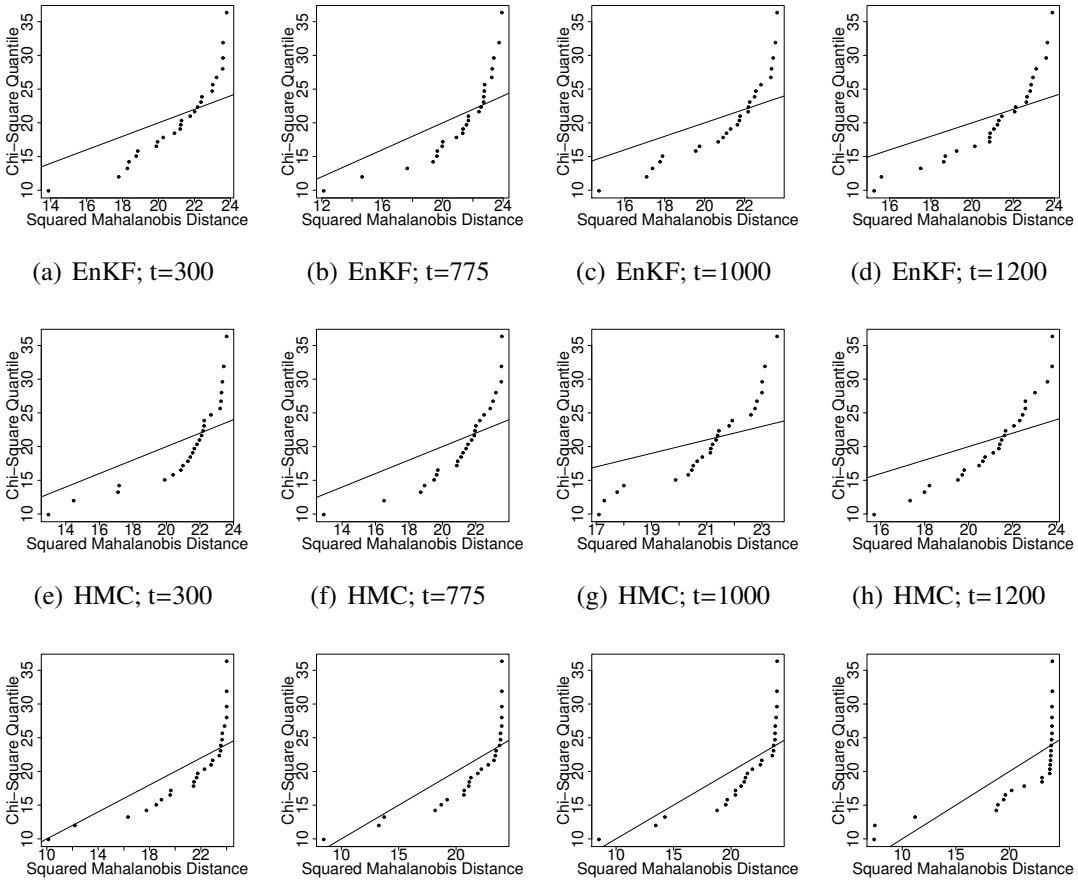
Results with nonlinear wind-magnitude observations

In the presence of a nonlinear observation operator the distribution is expected to show even stronger signs of non-Gaussianity. With stronger non-Gaussianity, the cluster methodology is expected to outperform the original formulation of the HMC sampling filter.

Figure 5.8 shows RMSE results, with the nonlinear observation operator, for the analyses obtained by HMC, $C\ell$HMC , MC-$C\ell$HMC filtering systems. While EnKF diverges under the current settings after the third cycle (results omitted for clarity), HMC, $C\ell$HMC , and MC-$C\ell$HMC continue to behave similar to the case where the linear observation operator is used.

Figure 5.9 shows rank histograms of HMC, $C\ell$HMC , and MC-$C\ell$HMC , with a nonlinear observation operator. We can see that $C\ell$HMC performance is similar to the case when the linear observation operator is used. It seems to be entrapped into a local minimum losing its dispersion quickly. The results of the MC-$C\ell$HMC filter avoid this effect and show a reasonable spread.

The results presented here suggest that the cluster formulation of the HMC sampling filter is advantageous, especially in the presence of highly nonlinear observation operator, or strong indication of non-Gaussianity.



(a) EnKF; t=300 (b) EnKF; t=775 (c) EnKF; t=1000 (d) EnKF; t=1200

(e) HMC; t=300 (f) HMC; t=775 (g) HMC; t=1000 (h) HMC; t=1200

(i) MC-$C\ell$HMC ; t=300 (j) MC-$C\ell$HMC ; t=775 (k) MC-$C\ell$HMC ; t=1000 (l) MC-$C\ell$HMC ; t=1200

Figure 5.7: Data assimilation with a linear observation operator. Chi-square Q-Q plots for the forecast ensembles obtained by the EnKF, HMC, and MC-$C\ell$HMC filtering systems at times t=300, 775, 1000, and 1200 provide strong indication of non-Gaussianity. The filtering methodology, and the assimilation time are given under each panel. Localization is applied to the ensemble covariance matrix to avoid singularity while evaluating the Mahalanobis distances of the ensemble members.

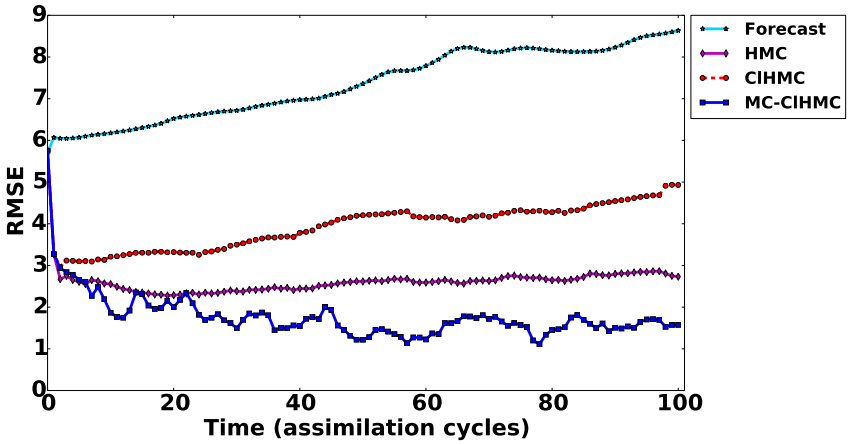


Figure 5.8: Data assimilation results with the nonlinear observation operator. RMSE of the analyses obtained by HMC, $C\ell$HMC , and MC-$C\ell$HMC filtering schemes.

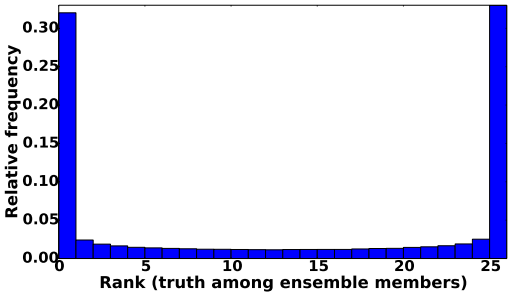
5.5 Conclusions and Future Work

This work presents a new formulation of the HMC sampling filter for non-Gaussian data assimilation. The new formulation, named the Cluster HMC sampling filter ($C\ell$HMC), relaxes the Gaussian prior assumption. The prior density is represented more accurately via a GMM fitted to the forecast ensemble. The initial formulation of the $C\ell$HMC filter presented here is not expected to outperform the original HMC filter unless the sampler is capable of efficiently sampling multimodal distributions with modes separated by large regions of low probability. A multi-chain version of $C\ell$HMC , namely MC-$C\ell$HMC , is developed in order to achieve this goal.

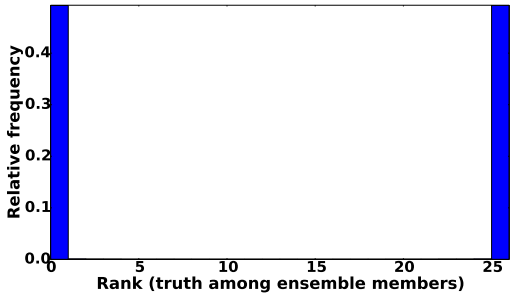
Numerical experiments are carried out using a nonlinear 1.5-layer reduced-gravity quasi geostrophic model in the presence of observation operators of different levels of nonlinearity. The results show that the new methodologies are much more efficient that the original HMC sampling filter especially in the presence of a highly nonlinear observation operator.

The MC-$C\ell$HMC is an algorithm that deserves further investigation. For example the local sample sizes here are selected based on the prior weight multiplied by the likelihood of the corresponding component mean. An optimal selection of the local ensemble size is required to guarantee efficient sampling from the target distribution.

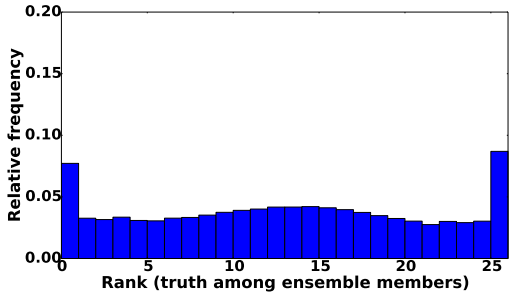
Instead of using MC-$C\ell$HMC filter, one can use $C\ell$HMC with geometrically tempered Hamiltonian sampler as recently proposed in [168], such as to guarantee navigation between separate modes of the posterior. Alternatively, the posterior distribution can be split



(a) HMC



(b) $C\ell$HMC +AIC



(c) MC-$C\ell$HMC +AIC

Figure 5.9: Data assimilation results using a nonlinear observation operator. The rank histograms of where the truth ranks among posterior ensemble members. The ranks are evaluated for every 16^{th} variable in the state vector (past the correlation bound) at 100 assimilation times. The filtering scheme used is indicated under each panel.

into N_c target distributions with different potential energy functions and associated gradients. This is equivalent to running independent HMC sampling filters in different regions of the state space under the target posterior.

Chapter 6

DATeS: A Highly-Extensible Data Assimilation Testing Suite

6.1 Introduction

Data Assimilation (DA) is research field of increasing interest for a wide range of applications and problems. DA refers to the fusion of information from different sources, including prior predictions of a numerical model, and snapshots of reality, in order to produce accurate description of the state of a physical system of concern [1, 2]. Applications of DA arise in many fields including geoscience, numerical weather forecasts, atmospheric composition predictions, oil reservoir simulations, and hydrology. Two approaches have gained wide popularity for solving the DA problems, namely ensemble and variational approaches. The ensemble approach is rooted in statistical estimation theory and uses an ensemble of states to represent the underlying probability distributions. The variational approach, rooted in control theory, involves solving an optimization problem to obtain a single "*analysis*" as an estimate of the true state of the system of concern. While the variational approach does not provide an inherent description of the uncertainty associated with the obtained analysis, it is less sensitive to physical imbalance prevalent in the ensemble approach. While the debate on which approach is better is ongoing, hybrid methodologies were designed to harnesses the best of the two worlds.

Numerical experiments are an essential ingredient in the development of any new data assimilation algorithm. The numerical experiments in the DA paradigm involve linear algebra routines, a numerical model implementation along with time integration routines,

and an assimilation algorithm. Current testing suites for DA applications are either too simplified or very general, many are tied to specific settings, and are usually completely written in a specific language. For a new idea to be tested with different models written in different languages, a researcher might have to rewrite the code for his/her algorithm, which is a tedious process. A unified testing suite for DA filtering and smoothing applications is important to enable researchers and students to explore different aspects of the DA process with minimal coding effort.

The DA research section (DAReS) at the National Center for Atmospheric Research (NCAR) provides DART [34] as a community facility for ensemble filtering. DART employs a modular programming approach to test different flavors of the ensemble Kalman filtering algorithm with operational models. DART is currently the most well founded, and maintained standard testing platform for ensemble-based Kalman filtering. It has a long history and is currently interfaced with the most interesting geophysical models. Moreover it gives access to practical, and well-established parallel algorithms. However, it is very general, as it can support operational settings, and it requires large learning overhead. The fact that DART is mainly written in Fortran makes it a very efficient testing platform, however this creates limitations especially when third party implementations are to be interfaced with DART.

Python is one of the most popular and powerful scripting languages that gives the power of reusing existing pieces of code via inheritance. Python is widely known to be a powerful scripting tool for scientific applications that can be used to glue legacy codes. This can be achieved by writing wrappers that can act as interfaces. Building wrappers around existing C, and Fortran code is a common practice in scientific research. Several automatic wrapper generation tools, such as SWIG [170] and F2PY [171], have been developed and are widely used to create proper interfaces between Python and low-level languages. Moreover, Python is virtually available on all GNU/Linux platforms which makes it a good candidate to work in both serial and parallel modes.

Despite Python and Fortran are both relatively easy to learn, it's much easier to develop advanced skills in Python than in Fortran. Compared to Fortran, in using Python one generally give up performance for productivity. The performance penalty in the scientific research is handled by delegating computationally intensive tasks to compiled languages such as Fortran. Fortunately, this approach is followed by the scientific Python modules such as Numpy, and Scipy. This means that Python code can be written to maintain both productivity, and efficiency in scientific coding.

In this work, we employ an object-oriented-programming approach to develop a unified and highly-extensible Python-based DA testing suite. The package is named DATeS , and

is intended to be an *open-source, work-in-progress* package for researchers, and learners interested in different aspects of DA, to use and contribute to. Students can use it as an interactive learning tool, and researchers can use it as experimental testing pad where they can focus on coding only their new ideas without worrying much about the other pieces of the DA process. The code developed by a researcher in the DATeS framework should fit with all other pieces in the package with minimal-to-no effort, as long as the programmer follows the "*flexible*" rules of DATeS .

This chapter is structured as follows. Section 6.2 describes the architecture of the package DATeS . In Section 6.3, we discuss how to work with DATeS elements, and setup a DA experiment. Section 6.4 describes a detailed experiment in DATeS framework. Conclusions and future design and development directions are discussed in Section 6.5.

6.2 DATeS Implementation

All DA applications and methodologies share common elements. For example, the majority of the ensemble filtering methodologies share nearly all the steps of the forecast phase, and even a considerable portion of the analysis steps. Moreover, all the DA applications involve common essential components such as linear algebra routines, model discretization schemes, and DA algorithms.

Throughout the past two decades DA has grasped the attention of researchers in the computational science community. As a result, different aspects of the DA problem have been implemented in different languages. For example, low-level languages such as Fortran and C have been (and are still being) extensively used to develop numerically efficient model implementations, and linear algebra routines. Both Fortran and C allow for efficient parallelization because these two languages are supported by common libraries designed for distributed memory systems such as MPI, and shared memory libraries such as Pthreads and OpenMP. To make use of these available resources and implementations, one has to either rewrite all the different pieces in the same programming language, or have proper interfaces between the different new and existing implementations.

The philosophy behind the design of DATeS is that "*a unified DA testing suite has to be open-source, easy to learn, and enable reusing and extending available code, in order to test new DA methodologies with minimal effort*". Consequently, the suite should allow for easy interfacing with external third-party code written in various languages such as linear algebra routines written in Fortran, assimilation routines written in C, or models written in C++. A unified testing suite should enable a researcher to focus on implementing his/her

own algorithm without worrying much about coding other pieces with optimally tuned settings. DATeS was developed with these goals in mind.

The rest of this section details several aspects of the core implementation of DATeS .

6.2.1 DATeS architecture

DATeS makes use of the fact that any numerical DA experiment involves the following components:

a) linear algebra routines,

b) a "forecast" computer model with spatial discretization routines,

c) a time integration methodology,

d) a DA methodology, e.g. a filter or a smoother.

In DATeS an independent set of modules is built for each of these essential components.

Linear algebra. The linear algebra routines are responsible for handling data structures representing model-based entities, such as model state vectors, observation vectors, and covariance matrices. For the linear algebra routines to be efficiently reusable, DATeS provides unified classes for each of the linear algebra data structures involved in DA applications. As all classes in DATeS , the linear algebra classes are implemented in Python, however the actual functionalities of the associated methods can be written either in Python, or in lower-level languages using proper wrappers. A class is implemented for each of the linear algebra data structures such as to enable updating, slicing, and manipulating an instance of the corresponding data structure. For example a model state vector class has to provide methods that enable accessing/slicing and updating entries of the state vector, a method for adding two state vector instances, and methods for applying specific scalar operations on all entries of the state vector such as evaluating the square root or the logarithm. Once an instance of a linear algebra data structure is created, all it's associated methods should be accessible via the standard Python dot operator.

Forecast model. We believe that the central role in a DA experiment is played by the model implementation. The dynamical model's functionality should be accessible by all

the other components of a DA application. Each model used in DATeS has to have an associated class providing methods to access it's supported functionalities. A unified class design should provide all essential tasks that can be carried out by the model implementation. Most of the existing third-party model implementations, especially in large-scale settings, have their own representations of the incorporated linear algebra data structures. For example a model written in C can represent a state vector in memory as a one dimensional C array with or without ghost cells. To propagate a model state forward in time, the model involves a time integration algorithm that is sometimes tied to the model implementation.

DATeS requires a model class to provide access to all the underlying linear algebra data structures, and time integration routines. For example, each model class has to provide a method `state_vector()` that creates an instance of a sate vector class. To propagate a model state forward in time, a model class has to provide a method `integrate_state()` that takes a state vector instance and time integration settings, and returns a trajectory (list of states) evaluated at requested future times. A base class is provided in DATeS with definitions of all the methods required to be supported by a model class.

While some linear algebra and the time integration routines are model-specific, DATeS also provides general-purpose linear algebra classes, and time integration routines that can be reused by newly created models with minimal effort by the developer.

Data assimilation process. Assimilation classes are build in DATeS in a way that enables maximal reusability via inheritance. DA algorithms manipulate model states and observations by applying widely used mathematical operations. Once a model object is passed to an instance of an assimilation class, the assimilation object has immediate access to the time integration routines, and all other model functionalities, via the model object. An instance of an assimilation class also has access to all model-based data structures, and the associated methods through the model instance using the Python dot operator.

An assimilation instance in DATeS , e.g., a filtering object, a smoothing object, or a hybrid object, is designed to carry out a single assimilation cycle. For example, in the filtering framework, an assimilation cycle refers to assimilating data at a single observation time by applying a forecast and an analysis step. On the other hand, in the smoothing context, several observations available at discrete time instances within an assimilation window are processed simultaneously in order to update the model state at a given time over that window. A smoothing class is designed to carry out the assimilation procedure over a single assimilation window.

In typical DA numerical experiments a data assimilation algorithm is applied on several consecutive cycles to test the long-term performance among other goals. We refer, hereafter, to the procedure of applying several assimilation cycles as the "assimilation process", in both the filtering and the smoothing contexts. Special classes in DATeS are provided to carry out these assimilation processes. A reference to an assimilation object must be added to the configurations of an assimilation process class. Moreover, the configurations of an assimilation process object have to specify appropriate timespans for an assimilation experiment, e.g. observation and assimilation timespans. Other configurations include observations (real or synthetic) to be assimilated at the assimilation time instances. Once an assimilation process object is instantiated, it has immediate access to both an assimilation object, and a model object through that assimilation object. The main functionality of an assimilation process object is to iterate the proper functionalities of the associated assimilation object over the consecutive assimilation cycles, based on the configurations of the assimilation process object.

A general description of DATeS layout is given in Figure 6.1. The enumeration in Fig-

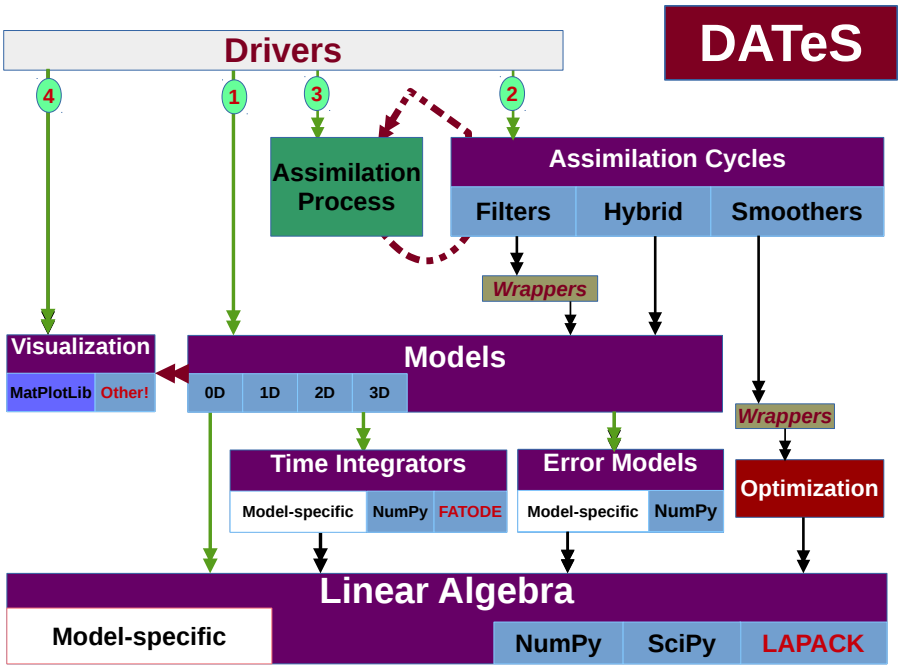


Figure 6.1: DATeS architecture.

ure 6.1 (numbers from 1 to 4 in green circles) indicates the proper order in which essential DATeS objects should be created. Specifically, one has to start with an instance of a model.

Once a model object is created, an assimilation object is instantiated, and the model object is passed to it. An assimilation process object is then instantiated, with a reference to the assimilation object passed to it. The assimilation process object iterates the consecutive assimilation cycles and save and/or output the results which can be optionally plotted later using visualization modules.

In the case where a single assimilation cycle is to be carried out, while one can only create a model object and an assimilation object, it is advisable to still create an assimilation process object and instruct it to carry out a single assimilation cycle by setting time parameters accordingly.

In DATeS all the involved components are independent such as to maximize the flexibility in experimental design. However, each newly added component must comply to DATeS rules to guarantee it's conformability with the other pieces in the package. DATeS provides base classes with definitions of the necessary methods. A new class added to DATeS , for example to implement a specific new model, has to inherit the appropriate model base class, and provide implementations of the inherited methods from that base class.

In order to maximize both flexibility and generalizability, we found that the best way to handle configurations, inputs, and output of DATeS object, is to use dictionaries. Parameters passed to instantiate an object are passed to the class constructor in the form of key-value pairs in dictionaries. We refer to these dictionaries, hereafter, as "*configuration dictionaries*". See Section 6.3 for examples on how to properly configure and instantiate DATeS objects.

Utility modules 6.2.7 are implemented in DATeS to provide auxiliary functionalities. As an example, the `configs` module provide functions for reading, writing, validating, and aggregating configuration dictionaries. In the ensemble approach of DA, an ensemble is a collection of state or observation vectors. In DATeS , an ensemble is represented by a list of either state, or observation vector objects. The utility modules include functions responsible for iterating over ensembles to evaluate ensemble-related quantities of interest, such as ensemble mean, ensemble variance/covariance, and even apply ensemble inflation.

In what follows we discuss the essential modules and classes in DATeS in detail.

6.2.2 Linear algebra classes

The main linear algebra data structures essential for almost all DA aspects are a) model state vector, b) state matrix, c) observation vector, and d) observation matrix. A state

matrix here refers to a square matrix of order equal to the model state space dimension. This matrix can be dense or sparse, and should not be constructed or saved in full, especially in large-scale applications. The effect of a state matrix on a state vector is of utmost importance in the DA context. For example a Jacobian-vector product is essential for evaluating the sensitivity matrix which is elementary to the solution the 4D-Var problem. An observation matrix here refers to a square matrix of order equal to the dimension of the observation subspace, and it's effect on an observation vector is also essential. Inverting an observation matrix is required by practical algorithms for solving the sequential filtering problem in the ensemble-based approach.

The entities discussed above are numerically manipulated by any DA methodology. For DATeS to be a unified framework for DA experimentation, a unified interface for accessing and manipulating these data structures is mandatory. Third-party linear algebra routines however can have radically different interfaces, and the underlying data structures in legacy and even new implementations are highly expected to be different. The idea in DATeS is to use unified Python classes for handling these linear algebra data structures.

A base class is provided in DATeS for implementations of each of these essential elements. In order to properly add a new class for a specific implementation of a given data structure, this new class must have the corresponding base class as a super class. Actual implementations of the methods in the subclass have to be provided for the inherited methods from the base class. **NotImplementedError** exception will be raised for methods without actual implementations in the derived class once they are invoked by other pieces in DATeS .

The following linear algebra base classes are provided in DATeS :

i) `state_vector_base.StateVectorBase`: a base class for state vector objects including all necessary methods,

ii) `state_matrix_base.StateMatrixBase`: a base class for state matrix objects with methods implementing necessary matrix operations,

iii) `observation_vector_base.ObservationVectorBase`: a base class for observation vector objects with related vector operations,

iv) `observation_matrix_base.ObservationMatrixBase`: a base class for observation matrix objects providing methods for related matrix operations.

Python descriptors must be provided in a linear algebra class to enable iterating a linear algebra data structure entries. Examples of these descriptors include `__getitem__()`,

`__setitem__()`, `__getslice__()`, `__setslice__()`, etc. These operators makes it feasible to standardize working with linear algebra data structures implemented in different languages or saved in memory in different forms. Given these descriptors, accessing any data structure can be carried out in Python code using the standard slice operator $[i,\ j : k,\ \ldots]$ where i, j, k are indexes of the entries in the underlying data structure.

DATeS provide actual Numpy-based classes to provide all necessary functionalities to linear algebra data structures represented as Numpy nd-arrays. The Numpy-based linear algebra classes in DATeS are:

i) `state_vector_numpy.StateVectorNumpy`

ii) `state_matrix_numpy.StateMatrixNumpy`

iii) `observation_vector_numpy.ObservationVectorNumpy`

iv) `observation_matrix_numpy.ObservationMatrixNumpy`

These classes provide guidance for designing more sophisticated extensions of the linear algebra classes.

6.2.3 Forecast model classes

Each model class in DATeS has to inherit the model base class: `models_base.ModelBase` or a class derived from it. A model class imports the proper linear algebra modules, for example a model implementation that uses Numpy nd-arrays to define it's associated linear algebra data structures, should import the four Numpy-based linear algebra classes mentioned in 6.2.2. The model class provides a method `state_vector()` that creates (or initializes) a state vector object, along with it's associated functionalities, i.e. it's methods. In general, this method either receives a reference to a data structure to be wrapped (for example a pointer to a one-dimensional `C` array) or called with no arguments in which case an instance of the imported state vector class is initialized with empty data structure representing the state vector associated to the model. Similar methods must be provided by the model class in order to properly initialize all the other linear algebra objects.

For a model to be useful, a time integration method has to be incorporated to propagate model state forward in time. Each model class has to provide implementation of the method `integrate_state()` which marches a state vector instance forward in time, based on the passed time integration parameters. This method returns a list of state vector

objects (a trajectory) evaluated at the passed timespan (e.g. an iterable passed to the input parameter `checkpoints`). An interested reader is referred the model base class implementation and documentation for further details on the necessary methods in a model class.

DATeS provides stable implementations of widely popular models for guidance including:

1. `lorenz_models.Lorenz3`: a class implementing the 3-variables Lorenz model [172].
2. `lorenz_models.Lorenz96`: an implementation of the Lorenz96 model (aka Lorenz-40) [68].
3. `cartesian_swe_model.CartesianSWE`: Cartesian shallow-water equations model [149, 150]. Model is written in C, with a SWIG wrapper.
4. `qg_1p5_model.QG1p5`: quasi-geostrophic (QG) model with double-gyre wind forcing and bi-harmonic friction [163]. Model is written in Fortran, with a F2Py wrapper.

6.2.4 Error models classes

It is not uncommon to follow a perfect model approach in DA applications, a case in which the model is generally deterministic rather than stochastic. However, some other sources of error cannot be ignored, for example background, and observation errors are essential in the formulation of nearly all DA methodologies.

We refer to the entity responsible for managing and creating noise vectors, sampled from a specific probability distribution function, as "*error model*". For example a Gaussian error model would be completely set up given the first and second order moments of the probability distribution it represents. `error_models_base.ErrorModelsBase` is the base class for error models in DATeS . This base class provides definitions for essential functionalities of an error model such as sampling the underlying probability distribution, and evaluating the value of the density function, e.g. upto a scaling factor.

The error models are basically related to model state, or the model observation vectors. This is mainly why DATeS require the model class to provide methods to create objects from the proper error model classes.

To initialize an error model, a model object has to be passed to the error model upon instantiation. Specifically, a model instance is passed to the error model as an option in

the error model configurations dictionary. Once an error model is created, it has means to create instances of a model state vector, an observation vector, a state matrix, or an observation matrix classes through the model. In DATeS , the error models are instantiated and attached to the model object upon instantiation of the model object itself.

In many practical DA applications, the errors are additive, and are modeled by random variables distributed according to a Gaussian distribution with zero mean, and a given or an unknown covariance matrices. DATeS implements a Numpy-based background, observation, and model error models respectively in the following classes:

1. `error_models_numpy.BackgroundErrorModelNumpy`
2. `error_models_numpy.ObservationErrorModelNumpy`
3. `error_models_numpy.ModelErrorModelNumpy`

These classes are derived from the base class `ErrorsModelBase`. These derived classes provide simplified methodologies to create second order statistics of the error variables, based on model trajectories, and the settings of the error model. The Numpy-based implementations of the error models can be used for guidance for extended error model classes. Please refer to the DATeS manual [173], or the website [174] for updated list of included models.

6.2.5 Assimilation classes

Assimilation classes are designed in dates to create assimilation objects responsible for carrying out a single assimilation cycle, and optionally print or write the results to files. For example an EnKF object should be designed to carry out a single assimilation cycle that consists of two steps, namely "forecast", and "analysis" steps. Unlike Models' classes, the methods associated with data assimilation objects are expected to be different. DATeS provides the common functionalities for filtering objects in the filtering base class `filters_base.FiltersBase`. All derived filtering classes should have the base filtering class as a super class. The basic assimilation objects in DATeS are a filtering object, a smoothing object, and a hybrid object. Here we focus on the design of filtering objects for simplicity and consistency. The interested reader is referred to the user manual [173] for further details.

As all the DATeS essential DA elements, a filtering object is instantiated by passing two configuration dictionaries to the class constructor. The filtering configurations including

the observation time, assimilation time, the observation vector, and the forecast state or ensemble, are passed in the first configuration dictionary `filter_configs`. An essential key in the filtering configuration dictionary is `model` which indicates a reference to a model object to be used in the filtering process. When a filtering object is successfully instantiated, it has immediate access to the model object and it's associated functionalities including error models, linear algebra structures, and time integration routines.

A `ValueError` exception will be raised if no model object is passed in the filtering configuration dictionary. The configurations related to filter output, to both screen and files, can be configures through the second argument (`output_configs`) of the filtering class constructor.

Here we want to clarify that the `filter_configs` configurations dictionary includes keys for both input and output parameters of the filtering object. For example it has two keys, `forecast_state` and `analysis_state`. Based on which of these is passed to the filter object upon instantiation, and based on another key (`forecast_first`) in the configuration dictionary, either `forecast_state` is considered as input, and `analysis_state` is viewed as output, or vice verse. Since the results of assimilation classes are expected to greatly differ, we opted to use the filter configurations dictionary as means of handling both filter configurations (including inputs), and filtering results.

The base class `filters_base.FiltersBase` has default configuration dictionaries for filtering and outputting. In a derived class, specific default configuration dictionaries should be provided and aggregated with the default configurations in the base class as explained in Section 6.3.

DATeS provides classes for several versions of the EnKF, the HMC family of filters, and a vanilla implementation of the particle filter:

1. `KF.KalmanFilter`: class implementing the standard Kalman filter equations [54, 53],
2. `EnKF.EnKF`: a class implementing the perturbed-observation (stochastic) EnKF [10, 11],
3. `EnKF.DEnKF`: a class implementing the deterministic EnKF [163],
4. `EnKF.ETKF`: a class implementing the ensemble transform Kalman filter (ETKF) [175],
5. `EnKF.LLSEnKF`: a class implementing the local least squares EnKF [176],
6. `hmc_filter.HMCFilter`: a class implementing the HMC sampling filter [36],

7. `multi_chain_mcmc_filter.MultiChainMCMC`: a class implementing the cluster HMC sampling filters ($C\ell$HMC , and MC-$C\ell$HMC) recently presented in [177],

8. `PF.PF`: a class implemented the vanilla particle filter [27].

These filtering classes can be instantiated and run with any of the DATeS model objects.

6.2.6 Assimilation process modules

A common practice in sequential DA experimental settings is to repeat a filtering cycle over a given timespan, with similar or different settings at each cycle. As mentioned in Subsection 6.2.1, we refer to this procedure as a "filtering process". To carry out a filtering process, one has to iterate the functionalities attached to a filter object. At each iteration, the filtering process object updates the filtering settings, such as the observation(s), the assimilation time, the observation time, whether the filtering is synchronous or asynchronous, and even update the true solution (if available). Other settings that a filtering process objects take care of is the output settings. For example, one might want to print results to screen or save them to files only after specific number of iterations. These operations can be done by creating an instance of the class `FilteringProcess` available in the module `filtering_process`. In fact the class `FilteringProcess` itself is a derived class from the super class `AssimilationProcess` available in module `assimilation_process_base`. The class `AssimilationProcess` is intended to be a base class for filtering, smoothing, and hybrid process classes.

As all classes discussed before, a class derived from `AssimilationProcess` should receive configuration dictionaries upon instantiation. A filtering process class constructor has two configuration dictionaries as parameters, namely `assimilation_configs`, and `output_configs`. The former is used to control the assimilation process over the consecutive filtering cycles. Essential keys in the `assimilation_configs` dictionary include a reference to the filtering object, observation and assimilation time instances, and observations whether synthetic or real. If the true solution is known, the synthetic observations can be obtained by perturbing the truth for example by adding noise generated by the observation error model attached to the model object which in turn is attached to the filter object. If a list of observations is passed to the filtering process object, the observations will be regarded as real observations, otherwise synthetic observations will be requested from the associated model object. For details on how to create a filtering process object see Subsection 6.3.3 and for more details see the user manual [173].

6.2.7 Utility modules

The module `dates_utility` provides supplementary functionalities to different aspects of DATeS . This module gives direct access to functions imported from several utility modules including:

1. `_utility_configs`: a set of functions to handle configuration dictionaries. This includes functions for aggregating, reading, and writing configuration dictionaries.
2. `_utility_stat`: provides functions to evaluate statistical quantities such as finding moments of an ensemble (e.g. list of model state or observation objects).
3. `_utility_machine_learning`: provides functions to carry out machine learning algorithms such as fitting a Gaussian Mixture Model to an ensemble.
4. `_utility_data_assimilation`: provides functions to carry out general DA tasks such as ensemble inflation, evaluating root-mean-squared errors (RMSE), and evaluating spatial decorrelation coefficient given the distance between two grid points and a localization function.

The utility module provides other general functions. For example, functions to handle file downloading, and functions for file I/O are also provided in other utility modules. For a list of all functions in the utility module, see the user manual [173].

Section 6.3 presents a discussion on how to properly create instances of the classes discussed here.

6.3 Using DATeS

One can use DATeS interactively, e.g. using IPython, or write a driver script to setup and run an assimilation experiment in DATeS framework. In both cases, the first step to run an experiment is to properly setup DATeS for a "run". Setting DATeS up for a run involves defining the root directory of DATeS as an environment variable, and adding the paths of DATeS source modules to the system path to be appropriately accessible. To set DATeS up for a run, navigate to the the root directory of DATeS , then run the following code snippet:

```
import dates_setup
dates_setup.Initialize_dates()
```

Snippet 6.1: initialize DATeS for a run.

A driver file can be placed in the root directory of DATeS or elsewhere. In the second case, however, the DATeS root path has to be manually added to the python system path, e.g. by adding `sys.path.insert()`, to the beginning of your script, where the standard Python module `sys` is imported first.

The rest of this section gives a brief discussion on how to correctly create create instances of the existing pieces in DATeS . We start by discussing how to create a model object, then a filter object, and finally a filtering process object.

6.3.1 Creating a model object

As mentioned before, a model class constructor has two configuration dictionaries, namely `model_configs`, and `output_configs`. Once the model class is imported, and these dictionaries are defined, the model object can be appropriately created. To create configuration dictionaries, a developer is referred to the user [173], or the help of the class constructor, for a list of supported keys and the associated default values. In general, if a class constructor receives unknown key, it will be ignored.

As an example, consider a 40-variables Lorenz96 with the configurations as follows: a) the forcing term is $F = 9.0$, b) a linear observation operator, and observation noise level 0.05 of the observation magnitude, c) only screen output and no file output, d) explicit Runge-Kutta time integration scheme. To create a Lorenz96 model object with these configurations, the following snippet can be used:

```
from lorenz_models import Lorenz96
model_configs = dict(force=9.0, time_integration_scheme='ERK')
output_configs = dict(file_output_output=False)
lorenz_model = Lorenz96(model_configs, output_configs)
```

Snippet 6.2: create an instance of Lorenz-96 model

Note that the state vector size (number of variables in the Lorenz96 model), the observation operator type, and the observation noise level are not specified in the `model_configs` dictionary here because these are the default values in the Lorenz96 class's model dictionary. The default values in the output configuration dictionary are used as well. To see the default values of all the configuration options, please see the DATeS manual [173], or see the the help of the Lorenz96 class.

Once the instance `lorenz_model` is created, all it's associated methods are accessible through that object. For example to initialize a state vector, one can use the command

`state = lorenz_model.state_vector()`. The methods attached to a state vector object can be immediately accessed through the created state vector instance `state`. For example to create a copy of `state` scaled by 0.4, one can use the command `scaled_state = state.scale(0.4, in_place=False)`, where the second argument, i.e. `in_place=False`, forces the `scale()` method to return a copy of `state` scaled by 0.4, while keeping `state` intact.

6.3.2 Creating a filter object

In order to create a filtering object, one has to create a model object first, and pass it to the filter's class constructor. Assume that `lorenz_model` object is already created as described in Subsection 6.3.1. To create an EnKF filtering object with ensemble size equal to 25, the following lines of code can be used:

```
initial_ensemble = lorenz_model.create_initial_ensemble(ensemble_size
    =25)

from EnKF import EnKF
filter_configs = dict(model=lorenz_model,
                      analysis_ensemble=initial_ensemble,
                      ensemble_size=25
                      )
output_configs = dict(file_output=False)
enkf_filter = EnKF(model_configs, output_configs)
```

Snippet 6.3: create an EnKF object

This creates an EnKF filtering object `enkf_filter`, however most of the methods associated to the EnKF object will raise exceptions if immediately invoked. This is mainly because several keys in the filter configuration dictionary are not appropriately assigned such as the observation, the forecast time, the analysis time, and the assimilation time. DATeS allows creating assimilation objects without these options to maximize flexibility. For example, a filtering process object could be completely responsible for updating the filter's configuration dictionary between consecutive filtering cycles. Here we use the model object to create an ensemble of 25 state vector sampled from the prior distribution (as specified by the default background error model attached to `lorenz_model` object). This ensemble is created by invoking the method `create_initial_ensemble()` attached to the model object. EnKF filter object requires an ensemble of model states to be assigned to either `analysis_ensemble` or `forecast_ensemble` keys in it's configurations dictionary. Based on the key to which the ensemble is assigned, and using both the time settings (e.g. `forecast_time, analysis_time`), and the flag `forecast_first` in the filter's

configuration dictionary, the filter object decides whether to carry out the forecast step or the analysis step first.

As mentioned earlier, in order to maximize flexibility, several options in the filter configuration dictionary are used for both input and output such as `analysis_ensemble`, `analysis_state`, `forecast_ensemble`, `analysis_state`, `forecast_state`.

6.3.3 Creating an assimilation process object

Let us assume we have created the model object `lorenz_model` as explained in Subsection 6.3.1, and the filtering object `enkf_filter` as explained in Subsection 6.3.2. Suppose we want to test EnKF with Lorenz96 model by repeating the assimilation cycle over a timespan from 0 to 10 with distances of 0.1 between each two consecutive observation/assimilation time. This can be implemented as follows:

```
import numpy as np
from filtering_process import FilteringProcess

obs_checkpoints = np.arange(0, 10.001, 0.1)
da_checkpoints = obs_checkpoints
ref_IC = lorenz_model.create_initial_condition()
assimilation_configs = dict(filter=enkf_filter,
                            obs_checkpoints=obs_checkpoints,
                            da_checkpoints=da_checkpoints,
                            forecast_first=True,
                            ref_initial_condition=ref_IC,
                            ref_initial_time=0
                            )
output_configs = dict(scr_output=True, scr_output_iter=1,
                      file_output=True, file_output_iter=5
                      )

experiment = FilteringProcess(assimilation_configs, output_configs)
experiment.recursive_assimilation_process()
```

Snippet 6.4: create a filtering process object to carry out EnKF filtering using Lorenz-96 model.

This will allow the filtering process object to modify the configurations of the filter object to print output to screen on each iteration (e.g. on each assimilation cycle), and save the results to file only after each assimilation cycles. The filter object itself provides methods responsible for screen and file output.

Given these minimal options in the dictionary `assimilation_config`, the filtering process object `experiment`, by design, will figure out the relevant configurations such as whether the assimilation is synchronous or not, whether to use real observations (if a list of observation is provided in the option `observation`) or to use synthetic observations created by perturbing the truth using the observation error model attached to the model object `lorenz_model`. Note that the true model state is assumed to be known for the synthetic Lorenz96 model here, with reference initial condition created by calling the method `create_initial_condition()` associate to the model object.

Once the filtering process object is created, the filtering experiment can be started by invoking the method `recursive_assimilation_process()` associated to the filtering process object.

Upon termination of a run in DATeS , executable files can be cleaned up by calling the utility function `clean_executable_files()` from the utility module.

The interested reader is referred to more detailed test cases available in the test_cases folder in the root directory of DATeS package source [174].

6.4 Example: $C\ell$HMC Experiment with QG Model

DATeS has been used to carry out the numerical experiments discussed in Chapter 5. Specifically, the numerical experiments discussed in 5.4 can be setup using slightly modified versions of the following driver script:

```
import dates_setup
import dates_utility as utility
import numpy as np

# Prepare DATeS for a run:
dates_setup.initialize_dates(random_seed=2345)

# Create a QG-1.5 model object:
from qg_1p5_model import QG1p5
conf = dict(MREFIN=5, model_name='QG-1.5',
            model_grid_type='cartesian',
            observation_operator_type='linear',
            observation_vector_size=50,
            observation_error_variances=4.0,
            observation_errors_covariance_method='diagonal',
            background_error_variances=5.0
            background_errors_covariance_method='diagonal',
```

```
            background_errors_covariance_localization_method='
    Gaspari_Cohn',
            background_errors_covariance_localization_radius=12
            )
model = QG1p5(model_configs=conf)

# Create ClHMC filter object:
ensemble_size = 25
initial_ensemble = model.create_initial_ensemble(ensemble_size=
    ensemble_size)

obs_checkpoints = np.arange(0, 1250.001, 12.5)
da_checkpoints = obs_checkpoints

from multi_chain_mcmc_filter import MultiChainMCMC
gmm_settings = dict(clustering_model='gmm',
                    cov_type='diag',
                    localize_covariances=False,
                    inf_criteria='aic',
                    number_of_components=None,
                    min_number_of_components=None,
                    max_number_of_components=None,
                    min_number_of_points_per_component=2,
                    invert_uncertainty_param=False,
                    approximate_covariances_from_comp=False,
                    use_oringinal_hmc_for_one_comp=False,
                    initialize_chain_strategy='forecast_mean'
                    )
mc_chain_params = dict(Symplectic_integrator='3-stage',
                       Hamiltonian_num_steps=20,
                       Hamiltonian_step_size=0.075,
                       Mass_matrix_type='prior_precisions',
                       Burn_in_num_steps=80,
                       Mixing_steps=10,
                       )
hmc_filter_configs = dict(model=model,
                          analysis_ensemble=initial_ensemble,
                          chain_parameters=mc_chain_parms,
                          prior_distribution='gmm',
                          gmm_prior_settings=gmm_settings,
                          ensemble_size=ensemble_size,
                          localize_covariances=False,
                          forecast_inflation_factor=1.0,
                          analysis_inflation_factor=1.0,
                          hybrid_background_coeff=1.0
                          )
```

```
hmc_output_configs = dict(scr_output=True,
                          file_output=False,
                          file_output_moment_only=False,
                          file_output_moment_name='mean',
                          file_output_separate_files=False
                          )

mc_mcmc_filter_configs = dict.copy(hmc_filter_configs)
mc_mcmc_filter_configs.update(dict(proposal_density='hmc',
                                   proposal_covariances_domain='local',
                                   gaussian_proposal_covariance_shape='
    diagonal',
                                   fix_gaussian_proposal_covar_diagonal
    =True,
                                   gaussian_proposal_variance = None,
                                   ensemble_size_weighting_strategy = '
    mean_likelihood',
                                   reduce_burn_in_stage=False,
                                   forced_num_chains = None,
                                   reset_random_seed_per_chain=True,
                                   random_seed_per_chain= 1,
                                   inter_comp_inflation_factor=1.0,
                                   intra_comp_inflation_factor=1.0
                                   )
                              )

mc_mcmc_output_configs = hmc_output_configs
filter_obj = MultiChainMCMC(filter_configs=mc_mcmc_filter_configs,
                            output_configs = mc_mcmc_output_configs
                            )

# Create filtering process object:
from filtering_process import FilteringProcess
ref_IC = model._reference_initial_condition.copy()
experiment = FilteringProcess(dict(model=model,
                                   filter=filter_obj,
                                   obs_checkpoints=obs_checkpoints,
                                   da_checkpoints=da_checkpoints,
                                   forecast_first=True,
                                   ref_initial_condition=ref_IC,
                                   ref_initial_time=0,
                                   random_seed=0
                                   ),
                              dict(scr_output=True,
                                   scr_output_iter=1,
                                   file_output=True,
```

```
                                        file_output_iter=1
                                        )
                                )
# Run the sequential filtering experiement:
experiment.recursive_assimilation_process()

# Clean executables and temporary modules:
utility.clean_executable_files()
```

Snippet 6.5: Running MC-$C\ell$HMC filter with QG-1.5 model and linear observation operator.

6.5 Conclusions

A flexible and highly-extensible testing suite, named DATeS , is developed in this dissertation and is described in this chapter. DATeS seeks to provide a unified testing suite for DA applications that allows researchers to easily understand and compare different methodologies in different settings with minimal coding effort. The core of DATeS is written in Python to enable for Object-Oriented capabilities. The main functionalities, such as model propagation, filtering, and smoothing code, can however be written in low-level languages such as C or Fortran to attain desirable levels of efficiency. While DATeS is in it's early stages, we plan to continue developing it over the next few years with the long-term goal of making it a complete DA testing suite for researchers and students.

Chapter 7

Conclusions and Future Research Directions

This dissertation develops a family of fully non-Gaussian data assimilation schemes, capable of handling nonlinearity that is ubiquitous in most applications such as weather forecasting, power-flow, medical image reconstruction, and seismology among others. The algorithms developed herein follow a Hamiltonian Monte-Carlo approach for sampling the posterior, and are capable of accommodating non-Gaussian probabilities, and nonlinearity in physical models and observation operators. While here we employ a Hamiltonian Monte-Carlo approach, the direct sampling routine can be easily replaced with other algorithms capable of sampling general probability densities.

The new ensemble-based data assimilation method, named "HMC sampling filter", obtains the analysis by sampling directly from the posterior distribution. This algorithm can be used for non-Gaussian filtering, and can also be used to replenish dead ensemble members in parallel implementation of the traditional filtering methodologies. We demonstrate the potential of this algorithm by applying it to the chaotic Lorenz-96 model in the presence of observation operators with different levels of non-linearity and differentiability. This methodology is also tested with shallow water model on a sphere with linear observation operator. The computational performance of the HMC sampling filter relies heavily on the choice of the pseudo-time parameters of the Hamiltonian trajectory built to propose states for the Markov chain. Future work will focus on automating the parameter tuning process in the context of large operational models at high-resolution.

The four-dimensional data assimilation smoother "HMC sampling smoother" works by sampling the posterior distribution at the initial time of an assimilation window. The HMC

smoother requires an adjoint of the simulation model, and runs on the same computational infrastructure as 4D-Var. The developed smoother is not limited to Gaussian errors or linear observations and can be applied to cases where other variational or ensemble methods may fail. Parallelizing the sampling smoother will be considered in Future work.

The original formulation of the HMC smoother works with the full dimensional model. It provides a consistent description of the posterior distribution, however it is very expensive due to the necessary large number of full model runs. This dissertation proceeds by incorporating reduced-order modeling to design efficient versions of the HMC sampling smoother, while retaining most of the accuracy of the full space HMC smoother. The reduced order HMC sampling smoothers employ reduced-order approximations of the model dynamics. A strong assumption made in the formulation of both the HMC sampling filter, and the HMC sampling smoother is that prior distribution can be well approximated using a Gaussian distribution. This assumption is made for simplicity, however it is a strong assumption since the forward propagation through nonlinear model dynamics will result in a non-Gaussian likelihood. In future work, we plan to investigate the possibility of using a KL-Divergence measure, developed in this work, between the high fidelity distribution and both the projected and the approximate posterior distribution, to guide the choice of the reduced-order subspace dimension. Multilevel HMC will be also considered to incorporate corrections obtained from several reduced-order simulations to enhance the quality of the reduced-order ensembles

The Gaussian-prior assumption made in the original formulation of the HMC sampling filter is relaxed by using Gaussian mixture models for prior density estimation. This dissertation proceeds by formulating a fully non-Gaussian filter, named the Cluster HMC sampling filter ($C\ell$HMC). The prior density is characterized more accurately by fitting a GMM to the prior ensemble. A multi-chain version of $C\ell$HMC , namely MC-$C\ell$HMC , is developed in order to force the filter to sample the vicinity of all probability modes in the formulated posterior. We test the performance of the clustering filters using a nonlinear 1.5-layer reduced-gravity quasi geostrophic model in the presence of observation operators of different levels of nonlinearity. The MC-$C\ell$HMC chooses the local ensemble size (size of ensembles collected bu each of the parallel chains), following a heuristic approach, based on the prior weight multiplied by the likelihood of the corresponding component mean. In the future we would like to investigate an optimal selection of the local ensemble size to guarantee efficient sampling from the target distribution.

The DATeS data assimilation testing suite is an extensible package with core functionality written in Python that follows an object-oriented-programming approach. DATeS separates the design of it's different components such as the linear algebra routines, the model, the assimilation algorithms, in order to maximize flexibility and usability. The devel-

oped package can interface existing third-party implementations written in C or Fortran. DATeS was used to carry out the numerical experiments involving $C\ell$HMC , MC-$C\ell$HMC sampling filters. While DATeS is still premature, we will continue developing it to serve as a unified testing environment for data assimilation research.

The family of algorithms proposed in this work provide a consistent framework for direct sampling from the posterior distribution in the data assimilation paradigm. In this work, a perfect model approach is followed. In order to broaden the range of applications of the sampling algorithms developed in this work, the proposed formulations can be extended in the future to accommodate model noise.

In this work, we followed a Hybrid Monte-Carlo sampling approach to sample the posterior distribution. The performance of the developed family of algorithms is greatly influenced by the choice of the parameters of the synthetic Hamiltonian system. For this family of algorithms to be applicable in operational settings, the parameters of the Hamiltonian trajectory should to be automatically selected such as to increase the acceptance rate, minimize the correlations between the selected ensembles, minimize the number of steps taken by the symplectic integrator and maximize the step size used to construct a Hamiltonian trajectory while maintaining stability. To achieve this goal, an interesting future research direction is to develop scalar optimization step(s) to tune the Hamiltonian parameters on the fly.

The cluster sampling filters, MC-$C\ell$HMC and MC-$C\ell$HMC , developed in this work are interesting, and deserve further investigation in the future. For example, the local sample sizes here are selected heuristically, e.g. based on the prior weight multiplied by the likelihood of the corresponding component mean in the prior mixture model. An optimal selection of the local ensemble size is required to guarantee efficient sampling from the target distribution. The MC-$C\ell$HMC filter was mainly proposed to guarantee that the $C\ell$HMC filter can sample all regions of high probability in the posterior. In the future, instead of using MC-$C\ell$HMC filter, one can investigate using $C\ell$HMC with geometrically tempered Hamiltonian (GTHMC) sampler, to guarantee navigation between separate modes of the posterior. Alternatively, the posterior distribution can be split into N_c target distributions with different potential energy functions and associated derivatives. This is equivalent to running independent HMC sampling filters in different regions of the state space under the target posterior.

In this work, we have investigated some strategies to reduce the cost of the proposed algorithms, while maintaining their accuracy, such as incorporating reduced-order models in the smoothing context. In order to apply these algorithms in operational settings, future work will address other strategies to reduce the associated computational cost. For exam-

ple in practical settings, the dimension of the observation subspace is always much smaller than that of the model subspace. In order to reduce the cost of the algorithms proposed in this work, an interesting future research direction is to sample the posterior distribution projected onto the observation subspace.

Apart from improving the family of algorithms developed herein, and developing new techniques, one of the main remaining future goals is to demonstrate the effectiveness of the developed family of algorithms in an operational setting.

Bibliography

[1] E. Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge University Press, 2002.

[2] R. Daley. *Atmospheric data analysis*. Cambridge University Press, 1991.

[3] Adrian Sandu and Tianfeng Chai. Chemical data assimilationan overview. *Atmosphere*, 2(3):426–463, 2011.

[4] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

[5] Jeffrey L Anderson. An ensemble adjustment kalman filter for data assimilation. *Monthly weather review*, 129(12):2884–2903, 2001.

[6] Craig H Bishop, Brian J Etherton, and Sharanya J Majumdar. Adaptive sampling with the ensemble transform kalman filter. part i: Theoretical aspects. *Monthly weather review*, 129(3):420–436, 2001.

[7] Edward Ott, Brian R Hunt, Istvan Szunyogh, Aleksey V Zimin, Eric J Kostelich, Matteo Corazza, Eugenia Kalnay, DJ Patil, and James A Yorke. A local ensemble kalman filter for atmospheric data assimilation. *Tellus A*, 56(5):415–428, 2004.

[8] M.K. Tippett, J.J. Anderson, and C.H. Bishop. Ensemble square root filters. *Monthly Weather Review*, 131:1485–1490, 2003.

[9] J. Whitaker and T. M. Hamill. Ensemble data assimilation without perturbed observations. *Monthly Weather Review*, 130:1913–1924, 2002.

[10] Burgers. G., van Leeuwen. P. J., and Evensen. G. Analysis scheme in the Ensemble Kalman Filter. *Monthly Weather Review*, 126:1719–1724, 1998.

[11] P.L. Houtekamer and H.L. Mitchell. Data assimilation using an ensemble Kalman filter technique . *Monthly Weather Review*, 126:796–811, 1998.

[12] A. Sandu and H. Cheng. A subspace approach to data assimilation and new opportunities for hybridization. *International Journal for Uncertainty Quantification*, In print, 2015.

[13] H. Cheng, M. Jardak, M. Alexe, and A. Sandu. A hybrid approach to estimating error covariances in variational data assimilation. *Tellus A*, 62(3):288–297, 2010.

[14] X. Wang, D. M. Barker, C. Snyder, and T. M. Hamill. A hybrid ETKF-3DVar data assimilation scheme for the WRF model. Part I: Observing system simulation experiment. *Monthly Weather Review*, 136(12):5116–5131, 2008.

[15] Hamill. T. M. and C. Snyder. A hybrid ensemble Kalman filter-3D variational analysis scheme. *Monthly Weather Review*, 128:2905–2919, 2000.

[16] Milija Zupanski, I Michael Navon, and Dusanka Zupanski. The maximum likelihood ensemble filter as a non-differentiable minimization algorithm. *Quarterly Journal of the Royal Meteorological Society*, 134(633):1039–1050, 2008.

[17] Pavel Sakov, Dean S Oliver, and Laurent Bertino. An iterative enkf for strongly nonlinear systems. *Monthly Weather Review*, 140(6):1988–2004, 2012.

[18] G. Evensen. The Ensemble Kalman Filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 53, 2003.

[19] T. M. Hamill and J. S. Whitaker. Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. *Monthly Weather Review*, 129:2776–2790, 2001.

[20] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forcast error statistics . *Journal of Geophysical Research*, 99(C5):10143–10162, 1994.

[21] P.L. Houtekamer and H.L. Mitchell. A sequential ensemble Kalman filter for atmospheric data assimilation . *Monthly Weather Review*, 129:123–137, 2001.

[22] Keston W Smith. Cluster ensemble kalman filter. *Tellus A*, 59(5):749–757, 2007.

[23] Milija Zupanski. Maximum likelihood ensemble filter: Theoretical aspects. *Monthly Weather Review*, 133(6):1710–1726, 2005.

[24] Ehouarn Simon and Laurent Bertino. Application of the gaussian anamorphosis to assimilation in a 3-d coupled physical-ecosystem model of the north atlantic with the enkf: a twin experiment. *Ocean Science*, 5(4):495–510, 2009.

[25] Jeffrey L Anderson and Stephen 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.

[26] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[27] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F-Radar and Signal Processing*, volume 140, pages 107–113. IET, 1993.

[28] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[29] Peter Jan Van Leeuwen. Particle filtering in geophysical systems. *Monthly Weather Review*, 137(12):4089–4114, 2009.

[30] M Ades and PJ Van Leeuwen. The equivalent-weights particle filter in a high-dimensional system. *Quarterly Journal of the Royal Meteorological Society*, 141(687):484–503, 2015.

[31] Peter Jan van Leeuwen. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, 136(653):1991–1999, 2010.

[32] Peter Jan Van Leeuwen. Efficient nonlinear data-assimilation in geophysical fluid dynamics. *Computers & Fluids*, 46(1):52–58, 2011.

[33] M. Fisher. How does 4D-Var handle nonlinearity and non-gaussianity? slides web site: http://4dvarenkf.cima.fcen.uba.ar/Download/Session_3/4DVar_nLnG_Fisher.pdf, 2008.

[34] Jeffrey Anderson, Tim Hoar, Kevin Raeder, Hui Liu, Nancy Collins, Ryan Torn, and Avelino Avellano. The data assimilation research testbed: A community facility. *Bulletin of the American Meteorological Society*, 90(9):1283, 2009.

[35] Lars Nerger, Wolfgang Hiller, and Jens Schröter. The parallel data assimilation framework pdaf - a flexible software framework for ensemble data assimilation. In *EGU General Assembly*, April 2012.

[36] A. Attia and A. Sandu. A hybrid Monte Carlo sampling filter for non-gaussian data assimilation. *AIMS Geosciences*, 1(geosci-01-00041):41–78, 2015.

[37] A. Attia, V. Rao, and A. Sandu. A sampling approach for four dimensional data assimilation. In *Dynamic Data-Driven Environmental Systems Science*, pages 215–226. Springer, 2015.

[38] Ahmed Attia, Vishwas Rao, and Adrian Sandu. A hybrid monte-carlo sampling smoother for four dimensional data assimilation. *International Journal for Numerical Methods in Fluids*, 2016. fld.4259.

[39] Ahmed Attia, Razvan Stefanescu, and Adrian Sandu. The reduced-order hybrid monte carlo sampling smoother. *International Journal for Numerical Methods in Fluids*, 2016. fld.4255.

[40] Yaqing Gu, Dean S Oliver, et al. An iterative ensemble kalman filter for multiphase fluid flow data assimilation. *Spe Journal*, 12(04):438–446, 2007.

[41] Eugenia Kalnay and Shu-Chih Yang. Accelerating the spin-up of ensemble kalman filtering. *Quarterly Journal of the Royal Meteorological Society*, 136(651):1644–1651, 2010.

[42] Chris Snyder, Thomas Bengtsson, Peter Bickel, and Jeff Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640, 2008.

[43] Shinya Nakano, Genta Ueno, and Tomoyuki Higuchi. Merging particle filter for sequential data assimilation. *Nonlinear Processes in Geophysics*, 14(4):395–408, 2007.

[44] M Jardak, IM 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.

[45] M Jardak, IM Navon, and M Zupanski. Comparison of ensemble data assimilation methods for the shallow water equations model in the presence of nonlinear observation operators. *Submitted to Tellus*, 2013.

[46] Alexandre Chorin, Matthias Morzfeld, and Xuemin Tu. Implicit particle filters for data assimilation. *Communications in Applied Mathematics and Computational Science*, 5(2):221–240, 2010.

[47] K. Law and A. Stuart. Evaluating data assimilation algorithms. *Monthly Weather Review*, 140:3757–3782, 2012.

[48] S. Duane, A.D. Kennedy, J. Brian B.J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[49] Andrew F Bennett and Boon S Chua. Open-ocean modeling as an inverse problem: the primitive equations. *Monthly weather review*, 122(6):1326–1336, 1994.

[50] Francis J Alexander, Gregory L Eyink, and Juan M Restrepo. Accelerated monte carlo for optimal estimation of time series. *Journal of Statistical Physics*, 119(5-6):1331–1345, 2005.

[51] SL Cotter, M. Dashti, and AM. Stuart. Variational data assimilation using targetted random walks. *International Journal for Numerical Methods in Fluids*, 68(4):403–421, 2012.

[52] S. E. Cohn. An introduction to estimation theory. *J. Meteorol. Soc. Jpn*, 75:257–288, 1997.

[53] R.E. Kalman. A new approach to linear filtering and prediction problems . *Transaction of the ASME- Journal of Basic Engineering*, 82:35–45, 1960.

[54] R E Kalman and R S Bucy. New results in linear filtering and prediction theory. *Journal of basic engineering*, 83(1):95–108, 1961.

[55] A.C. Lorenc. Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 112:1177–1194, 1986.

[56] Michael Fisher and Philippe Courtier. *Estimating the covariance matrices of analysis and forecast error in variational data assimilation*. European Centre for Medium-Range Weather Forecasts, 1995.

[57] R.M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. *Department of Computer Science, University of Toronto Toronto, Ontario, Canada*, 1993.

[58] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[59] Luke Tierney. Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728, 1994.

[60] R.M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2011.

[61] J.M. Sanz-Serna and M-P.Calvo. *Numerical Hamiltonian problems*, volume 7. Chapman & Hall London, 1994.

[62] J.M. Sanz-Serna. Markov chain Monte Carlo and numerical differential equations. In *Current Challenges in Stability Issues for Numerical Differential Equations*, pages 39–88. Springer, 2014.

[63] S. Blanes, F. Casas, and J-M. Sanz-Serna. Numerical integrators for the hybrid Monte Carlo method. *arXiv preprint arXiv:1405.3153*, 2014.

[64] A. Beskos, FJ. Pinski, J-M. Sanz-Serna, and A.M. Stuart. Hybrid Monte Carlo on Hilbert spaces. *Stochastic Processes and their Applications*, 121(10):2201–2230, 2011.

[65] M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[66] Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[67] Elias D Nino Ruiz, Adrian Sandu, and Jeffrey Anderson. An efficient implementation of the ensemble kalman filter based on an iterative sherman–morrison formula. *Statistics and Computing*, 25(3):561–577, 2015.

[68] Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.

[69] Edward N Lorenz and Kerry A Emanuel. Optimal sites for supplementary weather observations: Simulation with a small model. *Journal of the Atmospheric Sciences*, 55(3):399–414, 1998.

[70] Ahmed Attia and Adrian Sandu. A sampling filter for non-gaussian data assimilation. *arXiv preprint arXiv:1403.7137*, 2014.

[71] A. Beskos, N. Pillai, G. Roberts, J-M. Sanz-Serna, and A.M. Stuart. Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19(5A):1501–1534, 2013.

[72] Jeffrey L Anderson. A method for producing and evaluating probabilistic forecasts from ensemble model integrations. *Journal of Climate*, 9(7):1518–1530, 1996.

[73] O Talagrand, R Vautard, and B Strauss. Evaluation of probabilistic prediction systems. In *Proc. ECMWF Workshop on Predictability*, volume 1, page 25, 1997.

[74] M.D. Homan and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *The Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[75] A. St-Cyr, C. Jablonowski, J.M. Dennis, H.M. Tufo, and S.J. Thomas. A comparison of two shallow water models with nonconforming adaptive grids. *Monthly Weather Review*, 136:1898–1922, 2008.

[76] B. Neta, F.X. Giraldo, and I.M. Navon. Analysis of the Turkel-Zwas scheme for the two-dimensional shallow water equations in spherical coordinates. *Journal of Computational Physics, Elsevier*, pages 102–112, 1997.

[77] N. Gustafsson and J. Bojarova. Four-dimensional ensemble variational (4D-En-Var) data assimilation for the high resolution limited area model (HIRLAM). *Nonlinear Processes in Geophysics*, 21(4):745–762, 2014.

[78] B. R. Hunt, E. Kalnay, E. Kostelich, E. Ott, D. Patil, T. Sauer, I. Szunyogh, J. Yorke, and A. Zimin. Four-dimensional ensemble Kalman filtering. *Tellus A*, 56A(4):273–277, 2004.

[79] K. J. H. Law and A. M. Stuart. Evaluating data assimilation algorithms. *Monthly Weather Review*, 140(11):3757–3782, 2012.

[80] Raiul Toral and AL Ferreira. A general class of hybrid monte carlo methods. In *Proceedings of Physics Computing*, volume 94, pages 265–268, 1994.

[81] DM Ceperley, MH Kalos, and K Binder. Monte carlo methods in statistical physics, 1979.

[82] M. P. Vecchi S. Kirkpatrick, C. D. Gelatt. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[83] Y. Trémolet. Accounting for an imperfect model in 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 132(621):2483–2504, 2006.

[84] Y. Sasaki. Some basic formalisms in numerical variational analysis. *Monthly Weather Review*, 98(12):875–883, 1970.

[85] S Akella and I.M. Navon. Different approaches to model error formulation in 4D-Var: a study with high-resolution advection schemes. *Tellus A*, 61(1):112–128, 2009.

[86] Alexandru Cioaca and Adrian Sandu. An optimization framework to improve 4D-Var data assimilation system performance. *Journal of Computational Physics*, 275(0):377–389, 2014.

[87] F.X. Le Dimet, IM Navon, and D.N. Daescu. Second-order information in data assimilation. *Monthly Weather Review*, 130(3):629–648, 2002.

[88] M. Briers, A. Doucet, and S. Maskell. Smoothing algorithms for state-space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89, 2010.

[89] S. Ravela and D. McLaughlin. Fast ensemble smoothing. *Ocean Dynamics*, 57(2):123–134, 2007.

[90] G. Evensen and P.J. van Leeuwen. Assimilation of Geosat altimeter data for the Agulhas Current using the ensemble Kalman filter with a quasigeostrophic model . *Monthly Weather Review*, 124:85–96, 1996.

[91] G. Evensen and P.J. van Leeuwen. An ensemble Kalman smoother for nonlinear dynamics . *Monthly Weather Review*, 128:1852–1867, 2000.

[92] Elias D Nino Ruiz and Adrian Sandu. A derivative-free trust region framework for variational data assimilation. *Computational and Applied Mathematics*, 293:164–179, 2016.

[93] David J Earl and Michael W Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

[94] Robert H Swendsen and Jian-Sheng Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57(21):2607, 1986.

[95] I.M. Navon and Jian Yu. Exshall: A Turkel-Zwas explicit large time-step FORTRAN program for solving the shallow-water equations in spherical coordinates. *Computers and Geosciences*, 17(9):1311 – 1343, 1991.

[96] I. M. Navon and R. De Villiers. The application of the Turkel-Zwas explicit large time-step scheme to a hemispheric barotropic model with constraint restoration. *Monthly weather review*, 115(5):1036–1052, 1987.

[97] Anthony D'Augustine and Adrian Sandu. MATLODE: A MATLAB ODE solver and sensitivity analysis toolbox. *ACM Transactions on Mathematical Software (TOMS)*, Submitted, 2015.

[98] D.M. Dunlavy, T.G. Kolda, and E. Acar. Poblano v1. 0: A MATLAB toolbox for gradient-based optimization. *Sandia National Laboratories, Tech. Rep. SAND2010-1422*, 2010.

[99] Vishwas Rao and Adrian Sandu. A posteriori error estimates for variational inverse problems. *SIAM/ASA Journal on Uncertainty Quantification*, Submitted, 2014.

[100] Vishwas Rao and Adrian Sandu. A posteriori error estimates for DDDAS inference problems. *Procedia Computer Science*, 29:1256–1265, 2014.

[101] A. Attia, V. Rao, and A. Sandu. A Hybrid Monte-Carlo sampling smoother for four dimensional data assimilation. *arXiv preprint arXiv:1505.04724*, 2015.

[102] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[103] M.A. Grepl and A.T. Patera. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 39(01):157–181, 2005.

[104] A.T. Patera and G. Rozza. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations, 2007.

[105] G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15(3):229–275, 2008.

[106] M. Dihlmann and B. Haasdonk. Certified PDE-constrained parameter optimization using reduced basis surrogate models for evolution problems. *Submitted to the Journal of Computational Optimization and Applications*, 2013.

[107] 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.

[108] C.W. Rowley, I. Mezic, S. Bagheri, P.Schlatter, and D.S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.

[109] P.J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[110] G. Tissot, L. Cordir, N. Benard, and B. Noack. Model reduction using Dynamic Mode Decomposition. *Comptes Rendus Mécanique*, 342(6-7):410–416, 2014.

[111] D.A. Bistrian and I.M. Navon. An improved algorithm for the shallow water equations model reduction: Dynamic mode decomposition vs pod. *International Journal for Numerical Methods in Fluids*, 78(9):552–580, 2015.

[112] K. Karhunen. Zur spektraltheorie stochastischer prozesse. *Annales Academiae Scientarum Fennicae*, 37, 1946.

[113] M.M. Loève. *Probability Theory*. Van Nostrand, Princeton, NJ, 1955.

[114] H. Hotelling. Analysis of a complex of statistical variables with principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[115] E.N. Lorenz. Empirical Orthogonal Functions and Statistical Weather Prediction. Technical report, Massachusetts Institute of Technology, Dept. of Meteorology, 1956.

[116] L. Sirovich. Turbulence and the dynamics of coherent structures. I. Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987.

[117] L. Sirovich. Turbulence and the dynamics of coherent structures. II. Symmetries and transformations. *Quarterly of Applied Mathematics*, 45(3):573–582, 1987.

[118] L. Sirovich. Turbulence and the dynamics of coherent structures. III. Dynamics and scaling. *Quarterly of Applied Mathematics*, 45(3):583–590, 1987.

[119] S. Chaturantabut. Dimension Reduction for Unsteady Nonlinear Partial Differential Equations via Empirical Interpolation Methods. Technical Report TR09-38,CAAM, Rice University, 2008.

[120] S. Chaturantabut and D.C. Sorensen. A state space error estimate for POD-DEIM nonlinear model reduction. *SIAM Journal on Numerical Analysis*, 50(1):46–63, 2012.

[121] S. Chaturantabut and D.C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[122] Z. Drmac and S. Gugercin. A New Selection Operator for the Discrete Empirical Interpolation Method – improved a priori error bound and extensions, 2015.

[123] N.C. Nguyen, A.T. Patera, and J. Peraire. A 'best points' interpolation method for efficient approximation of parametrized function. *International Journal for Numerical Methods in Engineering*, 73:521–543, 2008.

[124] P. Astrid, S. Weiland, K. Willcox, and T. Backx. Missing point estimation in models described by Proper Orthogonal Decomposition. *IEEE Transactions on Automatic Control*, 53(10):2237–2251, 2008.

[125] K. Carlberg, C. Bou-Mosleh, and C. Farhat. Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

[126] K. Carlberg, R. Tuminaro, and P. Boggsz. Efficient structure-preserving model reduction for nonlinear mechanical systems with application to structural dynamics. preprint, Sandia National Laboratories, Livermore, CA 94551, USA, 2012.

[127] R. Everson and L. Sirovich. Karhunen-Loeve procedure for gappy data. *Journal of the Optical Society of America A*, 12:1657–64, 1995.

[128] R. Stefanescu and I.M. Navon. POD/DEIM Nonlinear model order reduction of an ADI implicit shallow water equations model. *Journal of Computational Physics*, 237:95–114, 2013.

[129] R. Stefanescu, A. Sandu, and I.M. Navon. Comparison of POD Reduced Order Strategies for the Nonlinear 2D Shallow Water Equations. *International Journal for Numerical Methods in Fluids*, 76(8):497–521, 2014.

[130] G. Dimitriu, I.M. Navon, and R. Stefanescu. Application of POD-DEIM Approach for Dimension Reduction of a Diffusive Predator-Prey System with Allee Effect. In *Large-Scale Scientific Computing*, pages 373–381. Springer, 2014.

[131] F. Fang, C.C. Pain, I.M. Navon, M.D. Piggott, G.J. Gorman, P.E. Farrell, P.A. Allison, and A.J.H. Goddard. A POD reduced-order 4D-Var adaptive mesh ocean modelling approach. *International Journal for Numerical Methods in Fluids*, 60(7):709–732, 2009.

[132] Y. Cao, J. Zhu, I.M. Navon, and Z. Luo. A reduced-order approach to four-dimensional variational data assimilation using Proper Orthogonal Decomposition. *International Journal for Numerical Methods in Fluids*, 53(10):1571–1584, 2007.

[133] PTM Vermeulen and AW Heemink. Model-reduced variational data assimilation. *Monthly Weather Review*, 134(10):2888–2899, 2006.

[134] K. Afanasiev and M. Hinze. Adaptive Control of a Wake Flow Using Proper Orthogonal Decomposition. *Lecture Notes in Pure and Applied Mathematics*, 216:317–332, 2001.

[135] S.S. Ravindran. Adaptive Reduced-Order Controllers for a Thermal Flow System Using Proper Orthogonal Decomposition. *Journal of Scientific Computing*, 23:1924–1942, 2002.

[136] K. Kunisch and S. Volkwein. Proper Orthogonal Decomposition for Optimality Systems. *Math. Modelling and Num. Analysis*, 42:1–23, 2008.

[137] R. Stefanescu, A. Sandu, and I.M. Navon. POD/DEIM reduced-order strategies for efficient four dimensional variational data assimilation. *Journal of Computational Physics*, 295:569–595, 2015.

[138] R. Stefanescu, I.M. Navon, H. Fuelberg, and M. Marchand. 1D+4D–Var Data Assimilation of lightning with WRFDA model using nonlinear observation operators. Technical report, Arxiv preprint: http://arxiv.org/abs/1211.2521, 2013.

[139] A. C. Antoulas. *Approximation of large-scale dynamical systems*. SIAM, Philadelphia, 2005.

[140] B.R. Noack, K. Afanasiev, M. Morzynski, G. Tadmor, and F. Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *Journal of Fluid Mechanics*, 497:335–363, 2003.

[141] R. Stefanescu, A. Sandu, and I.M. Navon. Comparison of POD reduced order strategies for the nonlinear 2D shallow water equations. *International Journal for Numerical Methods in Fluids*, 76(8):497–521, 2014.

[142] M. Barrault, Y. Maday, N. D. Nguyen, and A. T. Patera. An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique. Académie des Sciences*, 339(9):667–672, 2004.

[143] CG Khatri. Some results for the singular normal multivariate regression models. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 267–280, 1968.

[144] C.R. Rao. *Linear statistical inference and its applications*, volume 22. John Wiley & Sons, 2009.

[145] T.M. Cover and J.A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[146] M.B. Giles. Multilevel monte carlo path simulation. *Operations Research*, 56(3):607–617, 2008.

[147] A. Gregory, C. Cotter, and S. Reich. Multilevel ensemble transform particle filtering. *arXiv preprint arXiv:1509.00325*, 2015.

[148] R. Heikes and D.A. Randall. Numerical integration of the shallow-water equations on a twisted icosahedral grid. part i: Basic design and results of tests. *Monthly Weather Review*, 123(6):1862–1880, 1995.

[149] B. Gustafsson. An alternating direction implicit method for solving the shallow water equations. *Journal of Computational Physics*, 7:239–254, 1971.

[150] I. M. Navon and R. De-Villiers. Gustaf: A quasi-newton nonlinear ADI FORTRAN IV program for solving the shallow-water equations with augmented lagrangians. *Computers & Geosciences*, 12(2):151–173, 1986.

[151] Y. Saad. Sparsekit: a basic tool kit for sparse matrix computations. Technical Report, Computer Science Department, University of Minnesota, 1994.

[152] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[153] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Number 16 in Frontiers in Applied Mathematics. SIAM, 1995.

[154] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[155] I.M. Navon, X. Zou, J. Derber, and J. Sela. Variational Data Assimilation with an Adiabatic Version of the NMC Spectral Model. *Monthly Weather Review*, 120(7):1433–1446, 1992.

[156] J.R. Schott. A test for the equality of covariance matrices when the dimension is large relative to the sample sizes. *Computational Statistics & Data Analysis*, 51(12):6535–6542, 2007.

[157] Julian Besag and Peter J Green. Spatial statistics and bayesian computation. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 25–37, 1993.

[158] A Sokal. *Monte Carlo methods in statistical mechanics: foundations and new algorithms*. Springer, 1997.

[159] D. M. Higdon. Auxiliary variable methods for markov chain monte carlo with applications. *Journal of the American Statistical Association*, 93(442):585–595, 1998.

[160] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[161] Xuelei Hu and Lei Xu. Investigation on several model selection criteria for determining the number of cluster. *Neural Information Processing-Letters and Reviews*, 4(1):1–10, 2004.

[162] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[163] Pavel Sakov and Peter R Oke. A deterministic formulation of the ensemble kalman filter: an alternative to ensemble square root filters. *Tellus A*, 60(2):361–371, 2008.

[164] Ahmed Attia and Adrian. Sandu. Dates. http://csl.cs.vt.edu/software.php, 2016. [Online; accessed August 23, 2016].

[165] Gaspari G. and Cohn S.E. Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125:723–757, 1999.

[166] Reinhard Furrer, Marc G Genton, and Douglas Nychka. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3), 2006.

[167] G Candille and O Talagrand. Evaluation of probabilistic prediction systems for a scalar variable. *Quarterly Journal of the Royal Meteorological Society*, 131(609):2131–2150, 2005.

[168] Akihiko Nishimura and David Dunson. Geometrically tempered hamiltonian monte carlo. *arXiv preprint arXiv:1604.00872*, 2016.

[169] Kanti V Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3):519–530, 1970.

[170] David M Beazley et al. Swig: An easy to use tool for integrating scripting languages with c and c++. In *Tcl/Tk Workshop*, 1996.

[171] Pearu Peterson. F2py: a tool for connecting fortran and python programs. *International Journal of Computational Science and Engineering*, 4(4):296–305, 2009.

[172] E.N. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[173] Ahmed Attia, Ross Glandon, Paul Tranquilli, Mahesh Narayanamurthi, Arash Sarshar, and Adrian. Sandu. Dates: A highly-extensible data assimilation testing suite. https://bitbucket.org/a_attia/dates/wiki/Home, 2016. [Online; accessed August 23, 2016].

[174] Ahmed Attia, Ross Glandon, Paul Tranquilli, Mahesh Narayanamurthi, Arash Sarshar, and Adrian. Sandu. Dates: A highly-extensible data assimilation testing suite. https://bitbucket.org/a_attia/dates, 2016. [Online; accessed August 23, 2016].

[175] C. Bishop, B.J. Etherton, and S. Majumdar. Adaptive sampling with the Ensemble Transform Kalman Filter. Part I: Theoretical Aspects. *Monthly Weather Review*, 129:420–436, 2001.

[176] J.L. Anderson. A local least squares framework for ensemble filtering. *Monthly Weather Review*, 131(4):634–642, 2003.

[177] Ahmed Attia, Azam Moosavi, and Adrian Sandu. Cluster sampling filters for non-gaussian data assimilation. *arXiv preprint arXiv:1607.03592*, 2016.

Appendices

Appendix A

Appendix for a Hybrid Monte Carlo sampling filter for non-Gaussian data assimilation

A.1 Background and observation error covariances for Lorenz-96 experiments

The initial background error covariance matrix $\mathbf{B}_0$, and the observations error covariance matrix $\mathbf{R}$ used with Lorenz-96 model experiments are built as described below.

A.1.1 Initial background error covariance matrix $\mathbf{B}_0$

Let $\delta\mathbf{x} = \sigma_{\mathbf{x}_0}\, \mathbf{x}_0^{\text{ref}}$, where $\mathbf{x}_0^{\text{ref}}$ is the reference initial condition. $\sigma_{\mathbf{x}_0}$ is the noise level in initial background solution. The initial background error covariance matrix is given by:

$$\mathbf{B}_0 = 0.1 \times \mathbf{I} + \left(0.9 \times \left(\delta\mathbf{x}\,(\delta\mathbf{x})^T\right) \circ \rho\right) , \tag{A.1}$$

where $\mathbf{I}$ is the identity matrix, and ρ is the decorrelation matrix defined with localization radius $L = 4$. The first term is added to prevent zero or small variances from taking place.

The perturbation vector $\delta\mathbf{x}$ is given by:

$$\begin{aligned}\delta\mathbf{x} = [\quad & 0.2581,\ 0.2262,\ 0.2867,\ 0.4257,\ 0.6204,\ -0.0480,\\ & -0.0213,\ 0.4307,\ 0.2429,\ -0.3132,\ 0.1184,\ 0.3484,\\ & 0.6099,\ -0.1823,\ 0.1344,\ 0.3489,\ 0.6167,\ -0.3491,\\ & 0.5768,\ 0.1640,\ 0.0068,\ 0.4713,\ 0.3250,\ 0.0875,\\ & 0.3577,\ 0.6307,\ 0.4373,\ 0.1470,\ -0.0495,\ -0.1448,\\ & 0.0189,\ 0.5290,\ 0.2887,\ -0.1785,\ 0.2546,\ 0.5911,\\ & -0.1673,\ 0.2455,\ 0.6292,\ 0.7743\quad]^T,\end{aligned}$$

where the noise level is set to $\sigma_{\mathbf{x}_0} = 0.08$.

A.1.2 Observation error covariance matrices R

Observation error covariance matrices **R** are assumed to be diagonal and are created based on the observation operator in hand. Diagonals of **R** for the used observation operators are given as follows:

- Linear observation operator (2.21):

$$\begin{aligned}&[0.0273,\ 0.0271,\ 0.0263,\ 0.0326,\ 0.0314,\ 0.0258,\ 0.0283,\\ &\ 0.0273,\ 0.0323,\ 0.0287,\ 0.0294,\ 0.0340,\ 0.0223,\ 0.0281]^T.\end{aligned}$$

- Quadratic observation operator (2.22) with a threshold $a = 0.5$:

$$\begin{aligned}&[0.6901,\ 0.6022,\ 0.6442,\ 0.8984,\ 0.8009,\ 0.6371,\ 0.7297,\\ &\ 0.6929,\ 1.0260,\ 0.7944,\ 0.8087,\ 1.1770,\ 0.5506,\ 0.7371]^T.\end{aligned}$$

- Exponential observation operator (2.24) with factor $r = 0.2$:

$$\begin{aligned}&[0.0093,\ 0.0090,\ 0.0094,\ 0.0109,\ 0.0106,\ 0.0092,\ 0.0095,\\ &\ 0.0093,\ 0.0123,\ 0.0089,\ 0.0104,\ 0.0136,\ 0.0083,\ 0.0089]^T.\end{aligned}$$

- Exponential observation operator (2.24) with factor $r = 0.5$:

$$\begin{aligned}&[0.3096,\ 0.2065,\ 0.3227,\ 0.4626,\ 0.3911,\ 0.2820,\ 0.3281,\\ &\ 0.3266,\ 0.7467,\ 0.4050,\ 0.4228,\ 1.1328,\ 0.3087,\ 0.3206]^T.\end{aligned}$$

These observation error covariance matrices are created based on standard deviation of noise set to 5% of average magnitude of theoretical observations corresponding to the truth.

A.2 Symplectic numerical integrators

Here we present the five numerical integrators employed in this work. We start with the standard position Verlet integrator in A.2.1. The results of the standard Verlet are very sensitive to the choice of the time step. Three higher order integrators namely, two-stage (A.2.2), three-stage (A.2.3), and four-stage (A.2.4) position splitting integrators, are taken from [63]. These higher-order integrators lead to filters that are more stable and efficient than Verlet. The last integrator tested (A.2.5) is from [64] and is designed to work efficiently in infinite dimensional state spaces, and to avoid problems resulting from subtracting infinitely large numbers related to the total energy of the Hamiltonian system for infinite dimensional state spaces.

All the integrators are applied to a Hamiltonian system of the form (2.9) [62, 61].

A.2.1 Position Verlet integrator

One step of the position Verlet algorithm advances the solution of the Hamiltonian equations (2.9) from time t_k to time $t_{k+1} = t_k + h$ as follows [62]:

$$\begin{aligned} \mathbf{x}_{k+1/2} &= \mathbf{x}_k + \frac{h}{2}\mathbf{M}^{-1}\mathbf{p}_k\,, \\ \mathbf{p}_{k+1} &= \mathbf{p}_k - h\,\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_{k+1/2})\,, \\ \mathbf{x}_{k+1} &= \mathbf{x}_{k+1/2} + \frac{h}{2}\mathbf{M}^{-1}\mathbf{p}_{k+1}. \end{aligned}$$

The optimal time step h is $h \propto (1/\mathrm{N_{state}})^{1/4}$ [71]. The experiments show that the step size should be small (close to zero) to make this integrator stable. It may still fail for high dimensionality and whenever complications are present in the target distributions. The weakness of this simple integrator is illustrated in our numerical experiments with highly nonlinear observation operators.

A.2.2 Two-stage integrator

One step of the two-stage algorithm advances the solution of the Hamiltonian equations (2.9) from time t_k to time $t_{k+1} = t_k + h$ as follows [63]:

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_k + (a_1 h)\mathbf{M}^{-1}\mathbf{p}_k\,, & \mathbf{p}_1 &= \mathbf{p}_k - (b_1 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_1)\,,\\
\mathbf{x}_2 &= \mathbf{x}_1 + (a_2 h)\mathbf{M}^{-1}\mathbf{p}_1\,, & \mathbf{p}_{k+1} &= \mathbf{p}_1 - (b_1 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_2)\,,\\
\mathbf{x}_{k+1} &= \mathbf{x}_2 + (a_2 h)\mathbf{M}^{-1}\mathbf{p}_{k+1}\,,
\end{aligned}$$

where $a_1 = 0.21132$, $a_2 = 1 - 2a_1$, and $b_1 = 0.5$.

Linear stability analysis using a standard harmonic oscillator test problem [63] shows that the stability of this symplectic integrator is achieved for time step that lies in the interval $0 < h\omega < 2.6321480259$, where ω is the frequency of the oscillator.

A.2.3 Three-stage integrator

One step of the three-stage algorithm advances the solution of the Hamiltonian equations (2.9) from time t_k to time $t_{k+1} = t_k + h$ by the set of equations [63]:

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_k + (a_1 h)\mathbf{M}^{-1}\mathbf{p}_k\,, & \mathbf{p}_1 &= \mathbf{p}_k - (b_1 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_1)\,,\\
\mathbf{x}_2 &= \mathbf{x}_1 + (a_2 h)\mathbf{M}^{-1}\mathbf{p}_1\,, & \mathbf{p}_2 &= \mathbf{p}_1 - (b_2 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_2)\,,\\
\mathbf{x}_3 &= \mathbf{x}_2 + (a_2 h)\mathbf{M}^{-1}\mathbf{p}_2\,, & \mathbf{p}_{k+1} &= \mathbf{p}_2 - (b_1 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_3)\,,\\
\mathbf{x}_{k+1} &= \mathbf{x}_3 + (a_1 h)\mathbf{M}^{-1}\mathbf{p}_{k+1}\,,
\end{aligned}$$

where: $a_1 = 0.11888010966548$, $a_2 = 0.5 - a_1$, $b_1 = 0.29619504261126$, and $b_2 = 1 - 2b_1$.

Following the same argument as in A.2.2, the stability interval of the time step associated with this time integrator is of length ≈ 4.67, that is, h should be chosen such that $0 < h\omega < 4.67$, where ω is the frequency of the oscillator [63].

A.2.4 Four-stage integrator

One step of the four-stage algorithm advances the solution of the Hamiltonian equations (2.9) from time t_k to time $t_{k+1} = t_k + h$ as follows [63]:

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_k + (a_1 h)\mathbf{M}^{-1}\mathbf{p}_k\,, & \mathbf{p}_1 &= \mathbf{p}_k - (b_1 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_1)\,,\\
\mathbf{x}_2 &= \mathbf{x}_1 + (a_2 h)\mathbf{M}^{-1}\mathbf{p}_1\,, & \mathbf{p}_2 &= \mathbf{p}_1 - (b_2 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_2)\,,\\
\mathbf{x}_3 &= \mathbf{x}_2 + (a_3 h)\mathbf{M}^{-1}\mathbf{p}_2\,, & \mathbf{p}_3 &= \mathbf{p}_2 - (b_2 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_3)\,,\\
\mathbf{x}_4 &= \mathbf{x}_3 + (a_2 h)\mathbf{M}^{-1}\mathbf{p}_3\,, & \mathbf{p}_{k+1} &= \mathbf{p}_3 - (b_1 h)\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_4)\,,\\
\mathbf{x}_{k+1} &= \mathbf{x}_4 + (a_1 h)\mathbf{M}^{-1}\mathbf{p}_{k+1}\,,
\end{aligned}$$

where: $a_1 = 0.071353913450279725904$, $a_2 = 0.268458791161230105820$, $a_3 = 1 - 2a_1 - 2a_2$, $b_1 = 0.1916678$, and $b_2 = 0.5 - b_1$.

Again, as discussed in A.2.2, and A.2.3, this integrator has a stability interval $0 < h\omega < 5.35$ [63].

The time here has unspecified units. Generally speaking, the high order integrators (A.3, A.4, A.5), provide more favorable and wider stability ranges for the time step. For more on the stability intervals of the time step settings of these high-order integrators, see [63].

A.2.5 General integrator defined on Hilbert space

One step of the Hilbert integrator advances the solution of the Hamiltonian equations (2.9) from time t_k to time $t_{k+1} = t_k + h$ as follows [64]:

$$\begin{aligned}
\mathbf{p}_1 &= \mathbf{p}_k - \frac{h}{2}\mathbf{M}^{-1}\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_k)\,,\\
\mathbf{x}_{k+1} &= \cos(h)\mathbf{x}_k + \sin(h)\mathbf{p}_1\,,\\
\mathbf{p}_2 &= -\sin(h)\mathbf{x}_k + \cos(h)\mathbf{p}_1\,,\\
\mathbf{p}_{k+1} &= \mathbf{p}_2 - \frac{h}{2}\mathbf{M}^{-1}\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}_{k+1}).
\end{aligned}$$

As with the standard position Verlet integrator the selection criterion of step size is not precisely defined, however, it is designed to work with finite (non-zero) steps in infinite dimensional settings. Numerical results presented in Section 2.5 show that with careful tuning this integrator provides satisfactory results.