METHODS OF ENSEMBLE DATA ASSIMILATION ON ADAPTIVE MOVING MESHES

Colin Guider

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Mathematics in the College of Arts and Sciences.

Chapel Hill
2019

Approved by:

Chris K. R. T. Jones

Amarjit Budhiraja

Alberto Carrassi

Katie Newhall

Christian Sampson

ABSTRACT

Colin Guider: Methods of Ensemble Data Assimilation on Adaptive Moving Meshes
(Under the direction of Chris K. R. T. Jones)

Numerical models solved on adaptive moving meshes have become increasingly prevalent in recent years. In particular, neXtSIM is a 2D model of sea-ice that is numerically solved on a Lagrangian mesh that does not conserve the number of mesh points. In this dissertation, we present two novel approaches to the formulation of ensemble data assimilation for models with this underlying computational structure. Specifically, we map ensemble members onto a common reference mesh, where the Ensemble Kalman Filter (EnKF) can be performed. Numerical experiments are carried out using 1D prototypical models: Burgers and Kuramoto-Sivashinsky equations, with both Eulerian and Lagrangian synthetic observations assimilated. One of the approaches is very effective, while the other is significantly less so.

We also present a novel approach in the formulation of the Local Ensemble Transform Kalman Filter (LETKF) on a conservative moving mesh model. This is also achieved by mapping the ensemble members onto a common reference mesh, but it done in a significantly different manner than from the previous two approaches. Specifically, the common mesh is formed by taking an equidistributing mesh for the previous output of the algorithm. The preliminary results of this method from an application to Burgers equation are encouraging.

To Mom, Dad, Lauren, Sam, Neil, and Maddie

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Chris Jones. He took me on as a student at a time when I was unsure of which direction I wanted to take in my research, and gave me a very interesting problem that I think has real-world applicability. Chris stuck with me through my ups and downs, and I really appreciate that.

I would also like to thank my collaborators: Alberto Carrassi, Ali Aydogdu, Matthias Rabatel, and Pierre Rampal from the Nansen Center; Cassidy Krause and Erik Van Vleck at the University of Kansas; Nikhil Shankar at the University of Michigan; and John Maclean and Christian Sampson here at UNC - Chapel Hill. This work would not have been possible without them.

I would also like to acknowledge Colin Grudzien, Paul Cornwell, and Claire Kiers for great advice given to me along the way.

I want to thank everyone at the Nansen Center in Bergen, Norway for the warm hospitality and wonderful time I spent there. I would also like to thank everyone here in the math department here at UNC.

I would like to thank Chris, Alberto, Amarjit, Katie, and Christian for serving on my committee.

Finally, I would like to thank all of my family and friends for their support on this long journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Physical models are ubiquitous in environmental science [1, 2, 3]. They are key in our understanding of the underlying physical processes; from basic models that can be easily analyzed and understood, but are drastic simplifications of much more complicated physical phenomena, to much more complex ones, such as global climate models, that attempt to capture all aspects of the environment, but are much too large to fully understand. Of course, no matter how complex a model is, no matter how many parameters it has or how many equations are involved, it will never be able to perfectly represent the world we observe.

We then turn to the data for more information. With the the rise of big data in the past few decades, it has become more important than ever to be able to make use of the vast quantity of information available to us. This is a daunting endeavor: it is difficult to even comprehend the amount of data available. How can we hope to find even basic patterns in all of this, to find a signal in the noise? In many fields, like economics and political science, it is quite difficult to understand the underlying processes. There is a vast amount of data available, but it can be extremely difficult to find underlying trends.

The advantage in the physical sciences, particular in environmental science and climate, is that the physics involved is fairly well-understood. This, combined with the wealth of observations of various aspects of the atmosphere, suggests combining both of these approaches. We can begin with a physical model, usually a system of partial differential equations, that approximates some geophysical process like the greenhouse effect [2] or the evolution of a hurricane [4]. Of course, we have observations of many aspects of these two situations. For example, we have centuries worth of records of atmospheric carbon and global temperature measurements, and we can look back at hundreds of past hurricanes to track their paths, precipitation, and wind-speed. Our goal is to combine these data with our physical models to improve our knowledge of these phenomena. Generally, the objective is to use past and current observations to better predict what will happen

in the future, whether to project the global average temperature given various emissions scenarios or to make an informed decision as to which areas to evacuate given a hurricane's current trajectory. The field that combines physical models with observations is known as *data assimilation*.

1.1 The Field of Data Assimilation

We give a more thorough description of the basic techniques of data assimilation in Chapter 2, but we give a brief overview here. Generally, the goal is to estimate the value of some *state variable* $\mathbf{x}$, that evolves over time. The problem is that the variable $\mathbf{x}$ is not observed, at least not directly. However, we do have some idea of how the state variable evolves over time. This is given by a *state evolution model*, denoted by $\mathcal{M}$, which can depend on time and is often represented by a computational solution of a partial differential equation (PDE) model.

While we do not observe $\mathbf{x}$, we do observe a variable $\mathbf{y}$, that has some relationship with $\mathbf{x}$. This relationship is represented by

$$\mathbf{y} = \mathcal{H}(\mathbf{x}), \tag{1.1}$$

where $\mathcal{H}$ is known as the *observation operator*; this can also depend on time.

A key assumption underlying most situations where data assimilation is applicable is the presence of noise, both in the state model and observational model. Typically, this noise is taken to be Gaussian with zero-mean. This is where uncertainty is introduced into the situation, and is what makes the problem non-trivial. The variance in each of these noise terms is often unknown, but can be approximated using an ensemble-based process. This is the approach we take throughout. The problem of estimating the state given observations is fairly straightforward when the state and observational model are linear; the state models we consider later are exclusively nonlinear. This raises an issue we can also mitigate using ensemble-based methods.

In the next chapter, we detail the basic probabilistic data assimilation algorithms in use. In particular, we derive the Kalman Filter, and the Ensemble Kalman Filter (EnKF), its ensemble-based version. It is the the EnKF that is the basis for the algorithms developed in later chapters of this work. We also include the particle filter and hybrid filter, for completeness.

1.2 Adaptive Mesh Refinement

Models of the atmosphere and of specific atmospheric processes are generally represented by systems of partial differential equations. These equations will almost never have analytic solutions,

so there is no function we can find into which we can plug in the relevant state, location, and time and which will output the desired quantity. We take a numerical approach to these models. Using finite difference or finite element methods, we approximate solutions to these systems at various points in our spatiotemporal domain. For example, Global Climate Models (GCMs) "grid up" the atmosphere of the Earth into several million evenly distributed points, and the model equations are solved numerically at these grid points.

Large models are extremely computationally expensive; they must be run on clusters of computers. This is due to the number of mesh points on which the equations must be solved, in addition to the quantity of variables and equations involved. It may be desirable to reduce the complexity of the model in some way to reduce the computational cost needed to effectively run these models. One approach is to simplify some of the underlying geophysical processes, which would reduce the number of variables and equations in the model. Another approach, the one that is of interest to us here, is to find a way to use fewer mesh points. In order to be successful in such a method, we should strive to place more mesh points in regions of large variation and fewer points in other areas of the domain. Since the regions of variation will change over time, this leads us to the idea of *mesh adaptivity*.

Some of the existing literature [5] examines the scenario in which the number of mesh points is fixed in time. The mesh point advection is governed by a computational PDE that the mesh point velocities must satisfy. However, mesh points are not removed or inserted. For our purposes, we will refer to such meshes as *conservative*, in the sense that the number of mesh points is conserved throughout time. The idea is that the density of mesh points is higher where there is more variation in the variable(s) of interest, and the points move throughout the process to regions of higher variation as needed. In the one-dimensional case (we do not consider higher dimensions here), this is accomplished by equidistributing some function $\rho(x)$, such that its integral between consecutive mesh points is kept constant. This equidistribution condition can be reformulated as a partial differential equation that the adaptive mesh points must satisfy.

We will also consider adaptive meshes where the number of mesh points is allowed to change over time; we refer to such meshes as *non-conservative*. In this scenario, the mesh points move over time, but not according to an equidistribution condition as in the conservative mesh case. This lack of equidistribution can lead to meshes that are significantly distorted, and thus are unsuitable for

solving the relevant equations. To rectify this, a *remeshing* process is implemented in regions of significant distortion in the mesh. In general, mesh points are removed when the distance between mesh points becomes too small, and mesh points are added when the distance between them becomes too large. This remeshing process does not preserve the total number of mesh points, a significant deviation from the conservative case.

1.3 The Problem at Hand

The goal of this dissertation is to develop data assimilation algorithms suitable for adaptive moving mesh models. Data assimilation in general is discussed at length in Chapter 2. Since observations are becoming more and more ubiquitous in this age of big data, this is a problem relevant to several scientific areas, particularly in the geosciences.

Two different situations are considered here. The first is that of a non-conservative adaptive moving mesh model, such as that employed in neXtSIM. The framework for a non-conservative mesh in one dimension is developed in Chapter 3. One approach for a data assimilation method is employed in Chapter 4, and another in Chapter 5. The second is that of a conservative adaptive moving mesh model. An application of the Local Ensemble Transform Kalman Filter in this scenario is developed and analyzed in Chapter 6. The main ideas and future directions of this research are explored further in Chapter 7.

CHAPTER 2

Data Assimilation

2.1 What is Data Assimilation?

The problem of data assimilation can be approached using variational methods or probabilistic methods. Here, we take a probabilistic approach, since we prefer to construct approximate densities using an ensemble-based method. In this approach, we will repeatedly apply Bayes' Theorem to a prior density for the model and likelihood function to get a posterior density for the model state. We represent the true state and the observations by a sequence of random variables $\{(X_n, Y_n)\}_{n=0}^{N}$ that describe a two-component Markov chain. Thus, at time instant t_n, X_n is the true state of the system and Y_n is the observation. We use lower case variables x_n, y_n to denote realizations of these random variables X_n, Y_n, respectively. Observations are only encountered as realizations of random variables, so these will always be denoted in lower case. The underlying state process $\{X_n\}_{n=0}^{N}$ is unobservable, so we must conduct all inference based on the observations $\{Y_n\}$.

Because the state and observation are indexed by time, it is natural to pursue an "online" data assimilation algorithm; i.e., one that updates the state estimate as new observations come along. We achieve this using a Bayesian approach.

It will be helpful to simplify the notation. We denote the sets of state variables and observations by $X_{0:n} = \{X_0, X_1, \ldots, X_n\}$ and $Y_{1:n} = \{Y_1, \ldots, Y_N\}$, respectively. We similarly define the realizations of these random variables by $x_{0:n}$ and $y_{1:n}$. Then an application of Bayes' Theorem gives

$$p\left(x_{0:n}|y_{1:n}\right) \propto p\left(y_{1:n}|x_{0:n}\right) p\left(x_{0:n}\right). \tag{2.1}$$

In accordance with the preferred nomenclature, $p\left(x_{0:n}|y_{1:n}\right)$ is referred to as the posterior density, $p\left(y_{1:n}|x_{0:n}\right)$ as the likelihood density of the observtions $y_{1:n}$ given the state $x_{0:n}$, and $p\left(x_{0:n}\right)$ as the prior density.

Repeated application of (2.1) would require updating the joint distribution of $X_{0:n}$ in order to

assimilate the observation y_n for each n. We would also have to calculate the joint likelihood of every observation $Y_{1:n}$. Rather than working with joint densities, which would be overly cumbersome, we formulate the *filtering* problem, in which natural assumptions regarding the structure of the model and obserations allows us to rewrite (2.1) so that we can recursively update the probability density function (pdf) for just the current state X_n at time instant t_n, instead of updating the joint density of $X_{0:n}$. We then consider more specialized scenarios: we first describe the general Kalman filter, and then a statistical method known as the Ensemble Kalman Filter (EnKF). Useful versions of the EnKF include the Stochastic EnKF and the Local Ensemble Transform Kalman Filter (LETKF), which are both used in the adaptive mesh scenarios discussed in later chapters. Then, we describe a basic formulation of the Particle Filter.

2.2 Filtering

We make two key assumptions, which are both justified and significantly simplify things. First, we assume that the distribution of Y_n given $\{Y_k\}_{k=0}^{n-1}$ and X_n depends only on X_n. Second, we assume that the distribution of X_n given X_{n-1} and $\{Y_k\}_{k=0}^{n-1}$ depends only on X_{n-1}. This dependency structure is shown in Figure 2.1. Given these assumptions, we can rewrite (2.1) as

$$p\left(x_n|y_{1:n}\right) \propto p\left(y_n|x_n\right) p\left(x_n|y_{1:n-1}\right), \tag{2.2}$$

where the forecast density is $p\left(x_n|y_{1:n-1}\right)$ and the posterior density is $p\left(x_n|y_{1:n}\right)$. The question of computing the posterior density $p\left(x_n|y_{1:n}\right)$ is the filtering problem, and methods that accomplish this task are called filtering methods or filters. In order to compute the forecast density $p\left(x_n|y_{1:n-1}\right)$, one must integrate the posterior density $p\left(x_{n-1}|y_{1:n-1}\right)$ from the previous time step, as follows:

$$p\left(x_n|y_{1:n-1}\right) = \int p\left(x_n|x_{n-1}\right) p\left(x_{n-1}|y_{1:n-1}\right) dx_{n-1}. \tag{2.3}$$

This is the probabilistic analogue of a forecast model in the case of noiseless state dynamics. Once the forecast density is computed, the posterior density is then:

$$p\left(x_n|y_{1:n}\right) = p\left(y_n|x_n\right) p\left(x_n|y_{1:n-1}\right). \tag{2.4}$$

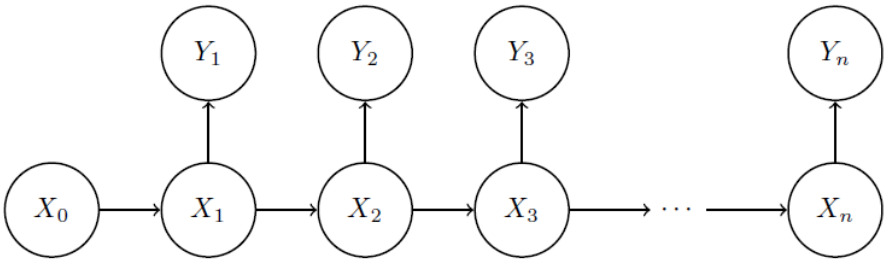


Figure 2.1: Dependency structure for nonlinear state space model

One advantage the probabilistic approach gives is that a posterior density is obtained, rather than simply a point estimate. Such estimates of what we term the *analysis* x_n^a can be given by the mean (or other measures of centrality such as the median and mode, depending on the context) of the posterior density $p(x_n|y_{1:n})$.

In the general case, we will not be able to produce a closed form expression for the integral in (2.4). Thus, we cannot obtain such expressions for the forecast and posterior distributions; this is because the state space models and observation functions will usually be nonlinear. It will be helpful to begin from a situation in which a closed-form expression can be derived, and from there progress to more sophisticated scenarios. In the most basic setting we consider, the filter we derive is known as the classical *Kalman Filter*.

2.3 Derivation of the Kalman Filter

Suppose that the state-observation Markov process had the dependence structure shown in Figure 2.1. Further, we suppose that the state model is linear with Gaussian errors; this means we can write it in the form

$$X_n \sim \mathcal{N}(\mathbf{M}_n x_{n-1}, \mathbf{Q}_n), \tag{2.5}$$

where $\mathbf{X}_n, x_n \in \mathbb{R}^M$ and $\mathbf{M}_n, \mathbf{Q}_n \in \mathbb{R}^{M\times M}$. In addition, we assume the observations are sampled independently and with Gaussian errors from the linear observation operator $\mathbf{H}_n$, with

$$Y_n \sim \mathcal{N}(\mathbf{H}_n x_n, \mathbf{R}_n), \tag{2.6}$$

where $Y_n \in \mathbb{R}^m$, $\mathbf{H}_n \in \mathbb{R}^{m\times M}$, and $\mathbf{R}_n \in \mathbb{R}^{m\times m}$. Namely, the conditional distribution of $(X_{0:n-1}, Y_{0:n-1})$ is normal with conditional mean $\mathbf{M}_n X_{n-1}$ and conditional variance $\mathbf{Q}_n$. Also,

the conditional distribution of Y_n given $(X_{0:n}, Y_{0:n-1})$ is normal with conditional mean $\mathbf{H}_n X_n$ and conditional variance $\mathbf{R}_n$. We assume that the initial state is normally distributed with some background covariance; that is, $X_0 \sim \mathcal{N}(0, \mathbf{B})$.

It is relatively straightforward to verify that the linear, Gaussian form of the model and observations imply that the prior, likelihood, and posterior densities in (2.1) are all Gaussian. This means that we can completely specify each of these distributions by their first and second moments, i.e., their mean and covariance. To further simplify the notation, we denote the mean of the prior density $p(x_n|y_{n-1})$ to be $x_{n|n-1}$ and its covariance by $\mathbf{P}_{n|n-1}$. Similarly, we denote the mean of the posterior density $p(x_n|y_{1:n})$ by $x_{n|n}$ and its covariance by $\mathbf{P}_{n|n}$. This notation refers back to the Bayesian formulation of the filtering problem. The first subscript represents the present time step, and the second subscript represents the time step of the most recent observation on which we condition. Using this notation in the model (2.5), we see that the forecast step in the data assimilation algorithm consists of updating the mean by way of the equation

$$x_{n|n-1} = \mathbf{M}_n x_{n-1|n-1} \tag{2.7}$$

and covariance

$$\mathbf{P}_{n|n-1} = \mathbf{M}_n \mathbf{P}_{n-1|n-1} \mathbf{M}_n^{\mathrm{T}} + \mathbf{Q}_n. \tag{2.8}$$

One can use (2.5) - (2.8), along with Bayes' Theorem, to determine that the posterior density is also Gaussian. Its mean and covariance are given by

$$\begin{aligned} x_{n|n} &= \mathbf{P}_{n|n} \left(\mathbf{H}_n^{\mathrm{T}} \mathbf{R}_t^{-1} + \mathbf{P}_{n|n-1}^{-1} x_{n|n-1} \right), \\ \mathbf{P}_{n|n} &= \left(\mathbf{H}_n^{\mathrm{T}} \mathbf{R}_t^{-1} \mathbf{H}_n + \mathbf{P}_{n|n-1}^{-1} \right)^{-1}. \end{aligned}$$

The mean can be rewritten as

$$x_{n|n} = x_{n|n-1} + \mathbf{K}_n \left(y_n - \mathbf{H}_n x_{n|n-1} \right), \tag{2.9}$$

while the covariance can be written as

$$\mathbf{P}_{n|n} = (I - \mathbf{K}_n \mathbf{H}_n)\, \mathbf{P}_{n|n-1}, \tag{2.10}$$

where I is the identity matrix and the matrix $\mathbf{K}_n$, known as the *Kalman gain*, is given by

$$\mathbf{K}_n = \mathbf{P}_{n|n-1} \mathbf{H}_n^{\mathrm{T}} \left(\mathbf{H}_n \mathbf{P}_{n|n-1} \mathbf{H}_n^{\mathrm{T}} + \mathbf{R}_n \right)^{-1}. \tag{2.11}$$

Equations (2.7) - (2.11) fully describe the forecast and analysis cycle for the Kalman filter at time step n.

2.4 The Ensemble Kalman Filter

The tractability and simplicity of the Kalman Filter has led to its widespread use, even in scenarios where the model conditions for its validity are not satisfied. One common example of such a setting is where the linear state model $\mathbf{M}_n x$ and linear observation operator $\mathbf{H}_n x$ are replaced by general nonlinear functions $m_n(x)$ and $h_n(x)$, respectively.

One approach to attempt to bypass linearization of the function m_n is what is known as the *Ensemble Kalman Filter (EnKF)*. In this case, one uses the full nonlinear state equation

$$x_n = m_n\left(x_{n-1}\right) + \tau_n \tag{2.12}$$

to simulate state values that are used to approximate the forecast density. More specifically, having obtained a Gaussian approximation for the posterior distribution at time step $n-1$, one takes L samples from this distribution, hereafter labeled as $\{X_{n-1}\}_{i=1}^{L}$. Using this and L realizations $\{\tau_n^i\}$ of the state noise, one uses the nonlinear state equation to produce the "forecast ensemble" $\{X_n^i\}$ according to the equation

$$X_n^i = m_n\left(X_{n-1}^i\right) + \tau_n^i, \tag{2.13}$$

where $i \in \{1, \ldots, L\}$. Assuming a Gaussian forecast density (which may not be valid in the case of a nonlinear model), we can use our forecast ensemble to approximate the mean of the forecast density p_n^f by the sample mean

$$x_{n|n-1} = \frac{1}{L} \sum_{i=1}^{L} X_n^i. \tag{2.14}$$

We approximate the forecast covariance by the ensemble covariance, given by the formula

$$\left(P_{n|n-1}\right)_{ij} = \frac{1}{L-1}\sum_{i=1}^{L}\left(X_n^i - x_{n|n-1}\right)\left(X_n^i - x_{n|n-1}\right)^{\mathrm{T}}. \tag{2.15}$$

The forecast distribution is used to produce a Gaussian posterior density by linearizing the observation function h_n and using a modified form of equations (2.9) - (2.10). An alternative to this is to use an "observation operator-free" version of the EnKF; this will prove useful in Chapter 4.

2.4.1 Stochastic Ensemble Kalman Filter

It turns out that simply using the ensemble-analogues of the Kalman Filter equations for the analysis step will result in the analysis covariance being underestimated by a factor of $(I - \mathbf{K_n}\mathbf{H_n})^{\mathrm{T}}$. The *stochastic Ensemble Kalman Filter* provides an adjustment to avoid this difficulty. An instance of the observational noise is inserted into the update step for each ensemble member. Thus, the update equations become

$$X_{n|n}^i = X_{n|n-1}^i + \mathbf{K}_n\left(y_n + e_n^i - \mathbf{H}_n X_{n|n-1}^i\right), \tag{2.16}$$

where $i \in \{1, \ldots, L\}$. The Kalman gain takes on the expected form, while we have

$$e_n^i \sim \mathcal{N}\left(0, \mathbf{R}_n\right).$$

As mentioned, adding these noise terms e_n^i ensures that the analysis distribution has the proper spread. If these noise terms are not added, then the analysis ensemble spread will be too small. We can then approximate the analysis covariance by

$$\mathbf{P}_{n|n} = \frac{1}{L-1}\sum_{i=1}^{L}\left(X_{n|n}^i - x_{n|n}\right)\left(X_{n|n}^i - x_{n|n}\right)^{\mathrm{T}}. \tag{2.17}$$

2.4.2 Local Ensemble Transform Kalman Filter

In Chapter 6, we will make use of the Local Ensemble Transform Kalman Filter (LETKF), introduced by [6]. The main characteristic of this algorithm is that it is a deterministic algorithm, unlike the stochastic EnKF. Domain localization is used here to reduce the dimensionality of the analysis. This is based on the (often reasonable) idea that the system is low-dimensional in

sufficiently small neighborhoods, and is primarily driven by the dynamics of nearby regions. Using this assumption, the analysis step is performed using different linear combinations of the ensemble members in different regions. In this way, a lower-dimensional, less computationally-intensive analysis can be used to explore a much higher-dimensional space.

The general idea is that the LETKF maps a background ensemble $\left\{X^i_{n|n-1}\right\}_{i=1}^{L}$ to an analysis ensemble $\left\{X^i_{n|n}\right\}_{i=1}^{L}$ by using local observations at time instant t_n. We consider an observation of this system as a triple $(y_n, h_n, \mathbf{R}_n)$, in which y_n is the observation, h_n is the (non-linear) observation operator, and $\mathbf{R}_n$ is the observation error covariance matrix. We assume observations have been truncated to only include those chosen for local analysis. We expect

$$y_n = h_n(x_n) + \epsilon_n,$$

where ϵ_n is a Gaussian random variable with distribution $\mathcal{N}(0, \mathbf{R}_n)$.

The goal is to infer which state x_n produced the observations, and take that as the analysis mean $x_{n|n}$. This can be rephrased as minimizing the following cost function

$$J(x_n) = \left[x_n - x_n^f\right]^{\mathrm{T}} \left(C_n^f\right)^{-1} \left[x_n - x_n^f\right] + \left[y_n - h_n(x_n)\right]^{\mathrm{T}} \mathbf{R}_n^{-1} \left[y_n - h_n(x_n)\right], \tag{2.18}$$

where $\left(C_n^f\right)^{-1}$ is the inverse restricted to the column space of C_n^f, which in turn requires that $x_n - x_n^f$ is an element of the column space [reference].

We use a modified version of the LETKF throughout our numerical experiments in Chapter 6. The modifications are designed to produce a useful definition of the analysis mean and covariance, given the utilization of adaptive mesh refinement.

2.5 The Particle Filter

To avoid relying on the linearity of the observation operator and/or of the state model, or to avoid using a linearity approximation, we can instead turn to a particle-based method. Here, we replace the high-dimensional integrals in (2.3) with suitable Monte-Carlo averages. Instead of using probability densities to describe the distributions we use discrete probability measures supported on finitely many points. These points and their wights will evolve in time to give the forecast measure Π_n^f and a posterior measure Π_n for different values of n One can then compute the analysis mean

x_n^a by computing an integral with respect to the measure Π_n.

2.5.1 A Basic Particle Filter

Suppose that Π_{n-1} is presented as a discrete probability measure supported on the points $\left\{X_{n-1}^1, \ldots, X_{n-1}^L\right\}$ with corresponding weights $\left\{p_{n-1}^1, \ldots, p_{n-1}^L\right\}$. Here, L represents the number of particles used in each step to approximate the distribution Π_{n-1}. The two key steps in the data assimilation algorithm proceed as follows:

1. (*Prediction Step*) We propagate each of the particles $X_{n-1}^i \to \hat{X}_n^i$ using the nonlinear state dynamics (2.12). This requires simulating L noise random varialbes τ_n^i, $i = 1, \ldots, L$. Given such random variables, the forecast ensemble members are defined by

$$\hat{X}_n^i = m_n\left(X_{n-1}^i\right) + \tau_n^i. \tag{2.19}$$

   Thus, we now have the forecast probability distribution Π_n^f as a discrete probability measure concentrated at L points $\left\{\hat{X}_n^i\right\}_{i=1}^L$ with weights $\{p_{n-1}^i\}_{i=1}^L$.

2. (*Filtering Step*) We update the weights $\{p_{n-1}^i\}_{i=1}^L$ using the observation y_n at time t_n by setting $p_n^i = cp_{n-1}^i\left(\hat{X}_n^i, Y_n\right)$, where

$$R(x,y) := \exp\left\{-\frac{1}{2}\left(y_n - h\left(\hat{X}_n^i\right)\right)^{\mathrm{T}} \mathbf{R}_n^{-1}\left(y_n - h\left(\hat{X}_n^i\right)\right)\right\}. \tag{2.20}$$

   The posterior distribution Π_n is then defined as the discrete measure with support points $\{X_n^i\}_{i=1}^L$ and weights $\{p_n^i\}$.

2.5.2 A Particle Filter with Resampling

The main idea here is to periodically resample from the discrete distribution Π_n, with replacement, to obtain a uniform distribution of weights. Of course, resampling introduces extra noise the approximating scheme, so it is important not to resample too frequently. We fix a resampling lag indicator $\alpha \in \mathbb{N}$. This parameter specifies the number of time steps between successive resampling steps. Suppose that Π_{n-1} is given as a discrete probability measure supported on points $\left\{X_{n-1}^1, \ldots, X_{n-1}^L\right\}$ with corresponding weights $\left\{p_{n-1}^1, \ldots, p_{n-1}^L\right\}$. If α does not divide n, then the posterior distribution is given exactly as before in (**??**). Otherwise, we further modify the discrete

probability measure Π_n. To do this, we take a random sample of size L from the discrete distribution

$$\left\{\left(X_n^1, p_n^1\right), \ldots, \left(X_n^L, p_n^L\right)\right\}$$

and relabel the new points as

$$\left(X_n^1, \ldots, X_n^L\right).$$

The posterior distribution is then given as the discrete distribution on the points $\left\{X_n^1, \ldots, X_n^L\right\}$ with all of the weights set equal to $1/L$.

2.6 Summary

In this chapter, we introduced the filtering problem and derived the forecast and analysis distributions given the prior and likelihood distributions. We did this using Bayes' theorem, which is fitting due to the probabilistic framework we have chosen. Since the integrals present in these distributions generally cannot be evaluated analytically, we turn to ensemble-based formulations of the data assimilation algorithms. In particular, we looked at a basic particle filter with resampling and two formulations of the Ensemble Kalman Filter: the stochastic EnKF and the LETKF.

One algorithm, not discussed in detail here, used by the author in work on inertial particle DA is the hybrid particle-ensemble Kalman filter, developed in [7]. The idea here is that certain models are both high dimensional and very nonlinear. The high dimensionality causes issues with a standard particle filter, while the nonlinearity and non-Gaussian posterior distributions can lead a standard EnKF to fail. A hybrid filter will take advantage of each of the individual methods where appropriate, while avoiding many of the pitfalls. It is shown in [7] that the hybrid filter performs much better than the EnKF, and about as well as the particle filter, with significantly fewer particles.

The stochastic EnKF is what we turn to in Chapter 4 and Chapter 5, while we make use of the LETKF in Chapter 6. These are the algorithms we modify to suit the adaptive moving mesh problems of interest. However, we must first develop the theory behind adaptive moving meshes, particularly those which do not conserve the number of mesh points. We do this in the next chapter.

CHAPTER 3

Physical Models on Adaptive Moving Meshes with Remeshing

3.1 Introduction

3.1.1 Adaptive mesh models

The computational model of a physical phenomenon is typically based on solving a particular partial differential equation (PDE) with a numerical scheme. Numerical techniques to solve PDEs evolving in time are most often based on a discretization of the underlying spatial domain. The resulting mesh is generally fixed in time, but the needs of a given application may require for the mesh itself to change as the system evolves, adapting to the underlying physics [8]. We consider here the impact of such a numerical approach on data assimilation.

Two reasons that may lead to the use of an adaptive mesh are: (1) for fluid problems, it is sometimes preferable to pose the underlying PDEs in a Lagrangian, as opposed to Eulerian, frame, or (2) the model produces a specific structure, such as a front, shock wave, or overflow that is localized in space. In case (1), the Lagrangian solver will naturally move the mesh with the evolution of the PDE [9]. For case (2), the idea is to improve computational accuracy by increasing the mesh resolution in a neighborhood of the localized structure (see, e.g. [10]). This may be compensated by the decrease in resolution elsewhere in the domain. Adapting the mesh can prove computationally efficient in that an adaptive mesh generally requires fewer points than a fixed mesh to attain the same level of accuracy [5]. Some important application areas where adaptive meshes have been used are: groundwater equations [11], and thin film equations [12], as well as large geophysical systems [13, 14].

3.1.2 Data assimilation for adaptive mesh models: the issue

Data assimilation (DA) is the process by which data from observations are assimilated into a computational model of a physical system. There are numerous mathematical approaches, and associated numerical techniques, for approaching this issue [15]. We use the term DA to refer to

the collection of methods designed to obtain an estimate of the state and parameters of the system of interest using noisy, usually unevenly distributed, data and an, inevitably approximate, model of its evolution [16]. There had been considerable development of DA methods in the field of the geosciences, particularly as a tool to estimate accurate initial conditions for numerical weather prediction models; a review of the state-of-the-art DA for the geosciences can be found in [17].

Mesh adaptation brings significant challenges to DA. In particular a time-varying mesh may introduce difficulties in the gradient calculation within variational DA [18]. In an ensemble DA methodology [19, 20], the challenge arises from the need to compare different ensemble members in the analysis step. With a moving mesh that depends on the initialization, different ensemble members may be made up of physical quantities evaluated at a different set of spatial points. There is another variation that is key to our considerations here, and that is relevant in both cases described above. The issue is that the nodes in the mesh may become too close together, or too far apart. Both situations can lead to problems with the computational solver. Some adjustment of the mesh, based on some prescribed tolerance, may then be preferable, and even necessary. We are particularly interested in the implications for DA when this adjustment involves the insertion or deletion of nodes in the mesh. The size of the mesh may then change in time, which has the consequence that the state vectors at different times may not have the same dimension. In other words, the state space itself is changing in dimension with time. Consequently, individual ensemble members, each of them representing a possible realization of the state vector, can even have different dimensions. In this situation, it is not possible to compute the ensemble-based mean and error covariances in a straightforward manner, a problem as these are at the core of the ensemble DA methods [19]. Dealing with and overcoming this situation is the main aim of this chapter and the following two chapters.

Two specific pieces of work can be viewed as precursors of our methodology. Bonan et al [21] study an ice sheet that is moving and modeled by a Lagrangian evolution, but without remeshing. The paper by Du et al. [22] develops DA on an unstructured adaptive mesh. Their mesh is adapted to the underlying solution to better capture localized structures with a procedure that is akin to the remeshing in neXtSIM. The challenge we address here is the development of a method that will address models that are based on Lagrangian solvers and involve remeshing.

3.1.3 Motivation: the Lagrangian sea-ice model *neXtSIM*

This work is further motivated by a specific application, namely performing ensemble-based DA for a new class of computational models of sea-ice [23]. In particular, the set-up we develop is based on the specifications of neXtSIM, which is a stand-alone finite element model employing a Maxwell elasto-brittle rheology [3, 24] to simulate the mechanical behavior of the sea ice. In this new rheological framework, the heterogeneous and intermittent character of sea ice deformation [25, 26] is simulated via a combination of the concepts of elastic memory, progressive damage mechanics, and viscous-like relaxation of stresses. This model has been applied to simulate the long-term evolution of the Arctic sea ice cover, with significant success when compared to satellite observations of sea ice concentration, thickness, and drift [24]. It has also been recently shown how crucial this choice for the ice rheology is in order to improve the model capabilities, e.g., to reproduce sea ice trajectories. This makes neXtSIM a powerful research numerical tool, not only to study polar climate processes, but also for operational applications; e.g., to assist search and rescue operations in ice-infested waters in the polar regions [27].

neXtSIM is solved on a 2-dimensional unstructured triangular adaptive moving mesh based on a Lagrangian solver that propagates the mesh of the model in time along with the motion of the sea ice [28]. Moreover, a mesh adaptation technique, referred to as *remeshing*, is implemented. It consists of a local mesh adaptation, a specific feature offered by the BAMG library that is included in neXtSIM (http://www.ann.jussieu.fr/hecht/ftp/bamg/bamg.pdf). The advantages of a local mesh modification is that it is efficient, introduces very low numerical dissipation [29], and also allows local conservation [30]. The remshing algorithm operates on the edges of the triangular elements to avoid tangling or distortion of the mesh, as well as inserting, or removing, nodes on the mesh in case it is needed to prevent exceedingly sharp refinements resulting in an excessive computational burden.

The specific DA methodology we develop for adaptive mesh problems is driven by the considerations of neXtSIM. The remeshing in neXtSIM, and the consequent change in the state vector's dimension, is addressed in our assimilation scheme by the introduction of a reference mesh. The latter represents a common mesh for forming the error covariance matrix from the ensemble members. The question then arises as to whether this common mesh is used to propagate each individual ensemble member forward in time. From the viewpoint of neXtSIM, however, continuing with

the reference mesh, common to all members, could throw away valuable physical information. In fact, the use of a Lagrangian solver in neXtSIM assures that the mesh configurations are naturally attuned to the physical evolution of the ice. For this reason, we make the critical methodological decision to map back to the meshes of the individual ensemble members after the assimilation step. The Lagrangian solver in the model is thus the primary determinant of the mesh configuration used in each forecast step. The reference mesh is only used in a temporary capacity to afford a consistent update at the assimilation step.

3.1.4 Goal and Outline

In this chapter, we proceed towards constructing a 1-dimensional setup designed to capture the core issues that neXtSIM presents for the applications of an ensemble-based DA scheme. We perform experiments using both Eulerian (where the observation locations are fixed) and Lagrangian (where observation locations move with the flow) observations. We test the strategy for two well-known PDEs: the viscous Burgers and Kuramoto-Sivashinsky equation, whose associated computational models we refer to as BGM and KSM, respectively. The Burgers' equation, which can be viewed as modeling a one-dimensional fluid, is a canonical example for which a localized structure, in this case a shockwave, develops and an adaptive moving mesh will get denser near the shock front. The Kuramoto-Sivashinsky equation exhibits chaotic behavior and this provides a natural test-bed for DA in a dynamical situation that is very common in physical science, and particularly in the DA applications in the geosciences (see Section 5.2 of [17]).

Our core strategy is to introduce a fixed reference mesh onto which the meshes of the individual ensemble members are mapped. A key decision is how refined the fixed reference mesh be made. There are two natural choices here: (a) one that has *at most* one node of an adaptive moving mesh in each of its cells, or (b) a reference mesh in which any adaptive moving mesh that may appear has *at least* one node in each cell of the fixed reference mesh. We refer to the former as a *high-resolution fixed referenced mesh* (HR) and the latter a *low-resolution fixed reference mesh* (LR). A natural guess would be that the high-resolution mesh will behave more accurately. Although this turns out to generally be ture, we will show that the low-resolution mesh may result in a better estimate when the ensemble is appropriately tuned.

There have been other recent studies aimed at tackling the issue of DA on adaptive and/or moving meshes. [31] studied a methodology to deal with a moving mesh model of an ice-sheet in a

variational DA framework. [21] extended the study and provided a comparison between a three dimensional variational assimilation (3D-Var) [] and an ensemble transform Kalman filter (ETKF) []. The mesh they use adapts itself to the flow of the ice-sheet but, in contrast to our case, the total number of nodes on the mesh is conserved.

[22] approach the issue in an ensemble DA framework using a three dimensional unstructured adaptive mesh model of geophysical flows [14, 32]. They adopt a fixed reference mesh on which the analysis is carried out. Each ensemble member is interpolated on a fixed reference mesh conservatively using a method called supermeshing developed in [33]. In [34], a similar methodology is used for a tsunami application which exploits adaptive mesh refinement on a regular mesh. Instead of using a fixed reference mesh, they use the union of meshes of all the ensemble members to perform the analysis. In summary, [21] addresses the issues that arise with a Lagrangian solver without any remeshing, whereas the approach in [22] is developed for a model that has the remeshing as part of its numerical algorithm, but uses an otherwise static mesh. The numerical solver underlying neXtSIM has both features and thus requires a methodology that differs from these two approaches. This work thus goes beyond existent works in developing a scheme that addresses the case of a moving mesh with non-conservative mesh adaptation.

In this chapter, we detail the problem of interest, describe the nature of the adaptive moving mesh methodologies in one dimension, and describe a remeshing process that is implemented intermittently. We also describe the model state and its evolution on the adaptive, non-conservative, 1D mesh. In Chapters 4 and 5, we introduce the EnKF using an adaptive moving mesh model in two different ways.

3.2 The physical model and its integration

This and the following chapters focus on a physical model describing the evolution of a scalar quantity u (e.g. the temperature, pressure, or one of the velocity components of a fluid) on a one-dimensional (1D) periodic domain $[0, L)$. We assume that a model of the temporal evolution of u is available in the form of a partial differential equation

$$\frac{\partial u}{\partial t} = f\left(u, \frac{\partial u}{\partial z}, \ldots, \frac{\partial^i u}{\partial z^i}, \ldots\right) \text{ where } i \in \mathbb{N},\ 0 \leq z \leq L,\ 0 < t_0 < t \tag{3.1}$$

with initial and boundary conditions

$$u(t_0, z) = u_0(z), \qquad u(t, 0) = u(t, L), \tag{3.2}$$

and with f being, in general, a nonlinear function. Realistic models of geophysical fluids incorporate (many) more variables, and expressed as a coupled system of PDEs. A notable example in the field of geosciences, and fluid-dynamics in general, is the system of Navier-Stokes equations; the fundamental physical equations in neXtSIM have the same form [24]. In this work, we consider the simpler 1D framework as a proxy of the 2D one in neXtSIM but, as will be made clear below, we formulate the 1D problem to capture many of the key numerical features of neXtSIM. Some of the challenges and issues for the higher dimensional case are discussed later.

Solving (3.1) numerically, with initial and boundary conditions (3.2), would usually involve the following steps: first, discretizing the original PDE in the spatial domain (e.g. using a central finite difference scheme), and then integrating, forward in time, the resulting system of ordinary differential equations (ODEs) using an ODE solver (e.g. an Euler or Runge Kutta method). The standard approach to numerically solving a PDE is appropriate when it is cast in an Eulerian frame. A key point about neXtSIM, however, is that it is solved in a Lagrangian frame. The use of a Lagrangian solver is a particular case of a class of techniques that is known as velocity-based methods in the adaptive mesh literature (see e.g. [9], and references therein). The dynamics of the adaptive mesh are given, in this case, by using u coming from the PDE (3.1) as the velocity field for the mesh points. [5] gives a comprehensive and detailed treatment of the case of adaptive meshes.

A further key feature of neXtSIM as a computational model is that it incorporates a remeshing procedure. As a result, it is different from the usual problems considered in the adaptive mesh literature [**?**]. In particular, it entails that, in general, no mapping exists from a fixed mesh to the adaptive mesh that is continuous in time. We call such an adaptive mesh *non-conservative* as the number of mesh points will change in time. It is this characteristic that we see as presenting the greatest challenge to a formulation of DA for neXtSIM, and addressing it in a model situation as the main contribution of this work, and one that makes it stand apart from previous work in the area of DA for computational models with non-standard meshes.

3.3 A one-dimensional, non-conservative, velocity-based adaptive moving mesh

3.3.1 The mesh features and its setup

We build here a 1D periodic adaptive moving mesh that retains the key features of neXtSIM's 2D mesh in being Lagrangian and including remeshing.

For a fixed time, a mesh is given by a set of points $\{z_1, z_2, \ldots, z_N\}$ with each $z_j \in [0, L)$. The z_j are referred to as the mesh nodes, or points, and we assume they are ordered as follows:

$$0 \leq z_1 < \cdots < z_j < \cdots < z_N < L. \tag{3.3}$$

To guide the remeshing, we define the notion of a *valid mesh* in which the mesh nodes are neither too close nor too far apart. To this end, we define two parameters: $0 < \delta_1 < \delta_2 < L$. A mesh $\{z_1, z_2, \ldots, z_N\}$ is a valid mesh if:

$$\delta_1 \leq |z_{j+1} - z_j| \leq \delta_2 \text{ for all } j \in \mathbb{N} : \ 1 \leq j < N-1, \quad \text{and} \quad \delta_1 \leq |_1 + L - z_N| \leq \delta_2. \tag{3.4}$$

It is further assumed that $\delta_2/\delta_1 \geq 2$ and that δ_2 and δ_1 are both divisors of L. The former hypothesis is related to the remeshing procedure and will be discussed in Section [], while the latter is useful in our DA approach and will be discussed in Section []. When condition (3.4) does not hold, the mesh is called an *invalid mesh*.

The mesh will evolve following the Lagrangian dynamics associated with the solution of the PDE (3.1). Each z_j will therefore satisfy the equation

$$\dot{z}_j = u(t, z_j), \tag{3.5}$$

where $\dot{=}\frac{d}{dt}$, and $u(t, z_j)$ is the velocity. The physical model (3.1), together with the mesh model (3.5), constitute a set of coupled equations that can be solved either simultaneously, or alternately [5]. In the former case, the mesh and physical models are solved at the same time, which strongly ties them together. A drawback of the simultaneous numerical integration is that the larger coupled system of equations arising by joining the mesh and the physical models is often more difficult to solve and may not conserve some features of the original physical model.

The neXtSIM model adopts an alternative strategy that bases the prediction of the mesh at time $t + \Delta t$, where Δt is the computational time step, on the mesh and the velocity field at the current time, t, and then subsequently obtain the physical solution on the new mesh at time $t + \Delta t$. As a consequence, the mesh is adjusted to the solution at one time-step earlier. This can cause imbalance, especially for low-resolution time discretization and rapidly changing systems, but it offers the advantage that the mesh generator can be coded as a separate module to be incorporated alongside the main PDE solver for the physical model. This facilitates the possible addition of conditions or constraints on the mesh adaptation and evolution. Having this ability is key to the remeshing procedure in neXtSIM.

In neXtSIM, the coupled system, which includes the mesh and the physical model, is solved in three successive steps: (1) The mesh solver is integrated to obtain the mesh points at $t + \Delta t$ based on the mesh and the physical solution at time t; (2) It is then checked whether the new mesh points satisfy the requisite condition and, if not, the *remeshing* procedure is implemented; (3) The physical solution is then computed at time $t + \Delta t$ on the (possibly remeshed) mesh at $t + \Delta t$.

In the first step, the movement of the mesh nodes is determined by the behavior of the physical model, which is a special case of the mesh being adaptive. In particular, the dynamics of the physical model can lead to the emergence of sharp fronts or other localized structures. These features can then be better resolved through the finer grid that now covers the relevant region, which is the usual motivation behind the use of adaptive meshes in general. This may result, however, in the allocation of a significant quantity of the total number of nodes to a small portion of the computational domain. Such a convergence of multiple modes in a small area can lead to a reduction of the computational accuracy in other areas of the model domain and to the increase of the computational cost as smaller time steps will be required. In the case of a mesh made up of triangular elements, as in neXtSIM, those may get too distorted, leading again to a reduction of the numerical accuracy of the finite element solution [35].

Adaptive mesh methods often invoke a mesh density function in (3.5) to control the mesh evolution [5]. In some cases, such as at a fluid-solid interface, large distortions may not be easily handled by moving mesh techniques alone, nor addressed by a mesh density function [36]. In these cases, a *remeshing* is performed (step (2) above) in order to distribute the nodes in the mesh consistently with the numerical accuracy and the computational constraints. In neXtSIM, an

analogous situation occurs due to the rheology that generates and propagates fractures or leads breaking the sea ice. For computational efficiency, a local remeshing is performed in the vicinity of a triangular element, called a cavity, when an element is too distorted. For example, [24] considers a triangular element distorted if it has a node with internal angle less than or equal to 10deg. The remeshing procedure involves adding new nodes and removing old ones if needed, as well as triangulation in the cavity to generate a suitable new mesh.

In the 1D models described later, the former challenge appears due to the nature of the physical system they describe. For instance, in Burgers' equation, the formation of a sharp shock-like front causes a convergence of mesh points. A suitable remeshing procedure is then applied.

We now view the mesh points $z_j = z_j(t)$ as evolving in time according to (3.5), and the computational time step Δt is chosen small enough so that the ordering given in (3.3) is preserved; the smallness of Δt has thus afforded the use of a low-order, straightforward Euler scheme to evolve the PDE forward in time. At each computational time step starting at, say, $t = t_k$, i.e., at each $t = t_k + i\Delta t$, remeshing may be performed according to the procedure given below.

3.3.2 The remeshing procedure

When an invalid mesh is encountered as a result of the advection process, a new valid mesh is created that preserves as many of these nodes as possible. A validity check is made at each computational time step. The remeshing is accomplished by looping through the nodes z_j at time t_k to check if the next node z_{j+1} satisfies (3.4) based on the parameters δ_1 and δ_2. Recall that we assume $\delta_2 \geq 2\delta_1$.

For each j, if the mesh node z_{j+1} is too close to z_j in that the left inequality in condition (3.4) is violated, then z_{j+1} is deleted. Similarly, if node z_{j+1} is too far from z_j, then a new node z^* is inserted in between z_j and z_{j+1} at $z^* = \frac{z_{j+1}+z_j}{2}$. Further, if $z_1 + L - z_N$ is either too large or too small, a similar procedure is implemented. We can understand now what motivates the choice of setting $\delta_2 \geq 2\delta_1$; if $\delta_2 \ll 2\delta_1$, then the insertion of a new node at the midpoint of the considered mesh points would create an invalid mesh.

The result of the remeshing will be a new mesh re-ordered according to (3.3) and the mesh will be valid in that (3.4) is satisfied. Note that any newly introduced node in the last step of the procedure may end up as either the first or last in the ordered set of mesh nodes. Further, a node that survives the remeshing may have a new index because of other new nodes or the deletion of

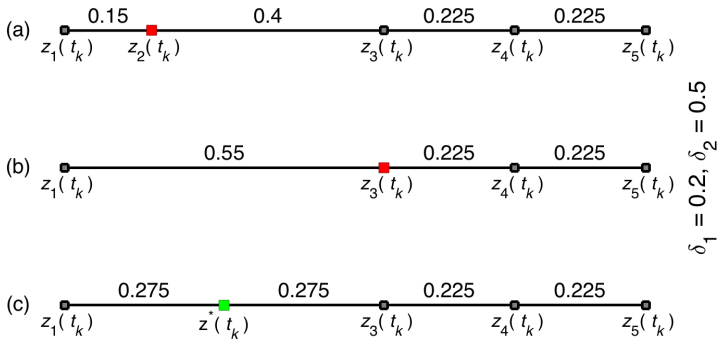


Figure 3.1: A simple illustration of the remeshing process with $\delta_1 = 0.2$ and $\delta_2 = 0.5$: invalid mesh **(a)** remove $z_2(t_k)$ which violates δ_1 **(b)** and insert $z^*(t_k)$ not to violate δ_2 **(c)**

old nodes. The number of nodes in a mesh may change after a remeshing. This has the implication that the dimension of the state vector will not be constant in time. It is this feature that makes the situation so different from standard DA and challenges us to create a new formulation.

The remeshing algorithm, with $\delta_1/\delta_1 = 0.2/0.5$, is illustrated in Figure 3.1, for the node $z_1(t_k)$ at a particular time $t = t_k$ of the integration. The node $z_2(t_k)$ has a distance of 0.15 from $z_1(t_k)$, which is smaller than δ_1: therefore, $z_2(t_k)$ is removed (Figure 3.1(a)). The next node, now $z_3(t_k)$, has distance 0.55 from $z_1(t_k)$, which exceeds δ_2 (Figure 3.1(b)): therefore a new node $z^*(t_k)$ is introduced at the midpoint between $z_1(t_k)$ and $z_3(t_k)$ (Figure 3.1(c)).

Figure 3.2 shows and example of this remeshing procedure applied to a velocity-based adaptive moving mesh using Burgers' equation (see SECTION for details) as a physical model. We see how the nodes, oriented along the horizontal axis, follow a moving front. In particular, the mesh, which initially has 40 equally distributed nodes ends up having only 27 unevenly distributed nodes, as a result of the remeshing procedure.

3.4 The model state and its evolution

Since both the physical value(s) representing the system and the mesh on which the PDE is solved are evolved, we represent them both in the state vector The dimension of the state vector is then $2N$, where N is the number of mesh nodes:

$$\mathbf{x} = (u_1, u_2, \ldots, u_N, z_1, z_2, \ldots, z_N) \in \mathbb{R}^N \times [0, L)^N, \tag{3.6}$$

where the z_i are viewed as the mesh nodes and u_i the values of the physical variable u at z_i.

The model will encompass all the algebraic relations of the computation, including the mesh

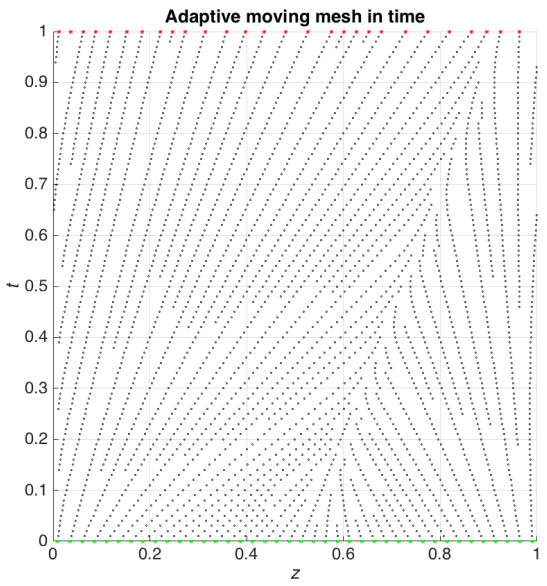


Figure 3.2: An illustration of adaptive moving mesh over time solving Burgers' equation until $t = 1$ on a domain $\mathbf{z}$=(0,1]. In this example, the remeshing criteria are based on $\delta_1 = 0.02$ and $\delta_2 = 0.05$. There are 40 initial adaptive moving mesh nodes and 27 at $t = 1$; these are shown in green and red, respectively.

advancement and remeshing. It will not be defined for every $\mathbf{x} \in \mathbb{R}^N \times [0, L)^N$. Indeed, the mesh nodes will need to satisfy (3.3). We therefore introduce $V_N \subset [0, L)^N$ by the condition that $\mathbf{z} = (z_1, z_2, \ldots, z_N) \in V_N$ when (3.3) holds.

The model will be operating between observation times. If we set $t = t_k$ as an observation time and $t = t_{k+1}$ as the next time at which observations will be assimilated, the model will be integrated with an adapting mesh, including Lagrangian evolution and possible multiple remeshings, from t_k to t_{k+1} If $\mathbf{x}_k = \mathbf{x}(t_k)$, then we set this model evolution as a map

$$\mathbf{x}_{k+1} = \mathcal{M}(\mathbf{x}_k). \tag{3.7}$$

Note that if the original PDE (3.1) is nonautonomous, i.e., f depends on t directly, then $\mathcal{M}$ will depend on k and we would write $\mathcal{M} = \mathcal{M}_k$. For convenience, we assume $t_{k+1} - t_k$ is a multiple of the computational time step. Moreover, we begin and end each integration between observation times with a remeshing if the given mesh is invalid. In this way, we guarantee that both $\mathbf{z}_k$ and $\mathbf{z}_{k+1}$ can be taken to correspond to valid meshes. In principle, we can then apply $\mathcal{M}$ to any element

$\mathbf{x} \in \mathbb{R}^N \times V_N$. Because of the tolerances of δ_1 and δ_2 there are, however, constraints on N. Since they are both divisors of L, we can introduce N_1 and N_2 by

$$L = N_1\delta_1 = N_2\delta_2, \tag{3.8}$$

and we can restrict $N_2 \leq N \leq N_1$. We can then view $\mathcal{M}$ as acting on a larger space that puts all of its possible domains together. To this end, we set $\mathbb{X}_N = \mathbb{R}^N \times V_N$ and, viewing each $\mathbb{X}_N$ as a distinct space, define

$$\mathbb{X} = \bigcup_{N=N_2}^{N_1} \mathbb{X}_N, \tag{3.9}$$

and cast $\mathcal{M}$ as a mapping from $\mathbb{X}$ to itself, i.e., $\mathcal{M} : \mathbb{X} \to \mathbb{X}$. We note again that N may change under this map, i.e., N may be different for $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$. In other words, if $\mathbf{x}_k \in \mathbb{X}_{N_k}$, then we will have for the next iteration $\mathbf{x}_{k+1} \in \mathbb{X}_{N_{k+1}}$ with, in general, $N_k \neq N_{k+1}$.

3.5 The Numerical Models

When testing the modified EnKF methodology described in the following chapters, we use two numerical models and two types of synthetic observations: Eulerian and Lagrangian.

The first numerical model is the diffusive form of Burgers' equation [1]

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial z} = \nu\frac{\partial^2 u}{\partial z^2}, \quad z \in [0,1), \quad t \in (0,T] \tag{3.10}$$

with periodic boundary conditions $u(0,t) = u(1,t)$. In our experiments, we set the viscosity $\nu = 0.08$; the model (3.10) is hereafter referred to as BGM. Given that Burgers' equation can be solved analytically, it has been used in several DA studies (see, e.g. Cohn, 1993; [37], [38]).

As a second model, we use an implementation of the Kuramoto-Sivashinsky equation ([39])

$$\frac{\partial u}{\partial t} + \nu\frac{\partial^4 u}{\partial z^4} + \frac{\partial^2 u}{\partial z^2} + u\frac{\partial u}{\partial z} = 0, \quad z \in [0,2\pi), \quad t \in (0,T], \tag{3.11}$$

which is also given periodic boundary conditions, and is referred to as KSM throughout this work. Concentration waves, flame propagation, and free surface flows are among the problems for which this equation is used. The higher order viscosity, ν, is chosen as 0.027 which makes (3.11) display chaotic behavior ([39]). Both numerical models are discretized using finite central differences and integrated

with an Eulerian time-stepping scheme. We integrate them using very small time-steps, 10^{-3} and 10^{-5} for BGM and KSM, respectively, since the equations are propagated forward explicitly.

Two "natural runs" are obtained, one for each model, by integrating them on a high resolution fixed uniform mesh. The size of the mesh for the natural run for BGM is 100 (corresponding to a resolution of 0.01), while it is 120 for KSM (equivalent to a resolution of about 0.052).

We have limited the time length of the simulations in BGM to $T = 2$ as the viscosity tends to dominate over longer times and dampen the wave motion. Figure 3.3a shows the natural run for BGM until $T = 2$ with the initial condition

$$u(z, 0) = \sin(2\pi z) + \frac{1}{2}\sin(\pi z). \tag{3.12}$$

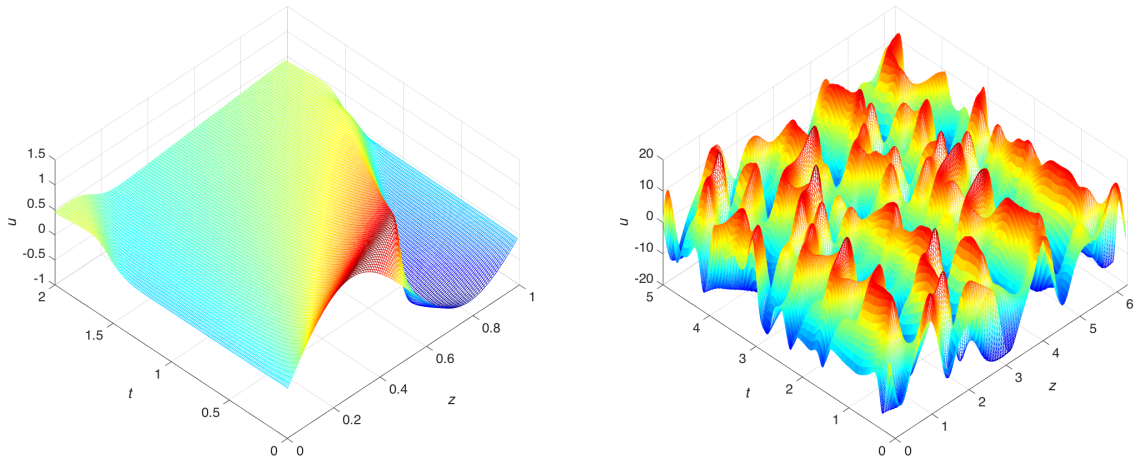
The figure shows clearly how the amplitude of the wave, picking around $z = 0.5$ for the initial time, is almost completely dampened out at the final time.

With the given choice of the viscosity, KSM is not as dissipative as BGM and simulations can be run for much longer. KSM is initialized using

$$u(z, 0) = -\sin(2\pi z) \tag{3.13}$$

as the initial condition. Then, it is spun-up until $T = 20$ and the solution at $T = 20$ is used as the initial condition for the DA experiments. Figure 3.3b shows the KSm natural run until $t = 5$ after re-initialization of the model following the spin-up (i.e. the actual simulation time being $T + t = 25$); the chaotic behavior of the KSM solution can be qualitatively identified by the random-like oscillations.

Synthetic Eulerian and Lagrangian observations are sampled from the natural run. Eulerian observations are always collected at the same, fixed-in-time, locations of the domain. We assume that Eulerian observers are evenly distributed along the one-dimensional domain (i.e. observations are at equally spaced locations) and their total number is constant, so the number of observations at time step t_k, $d^{EUL}(t_k) = d_k^{EUL} = d$ for all $k > 0$. The locations of the Lagrangian observations, on the other hand, change in time: the data are sampled by following the trajectories, solutions of the model. Being advected by the flow, Lagrangian observations may eventually concentrate within



(a) BGM (b) KSM

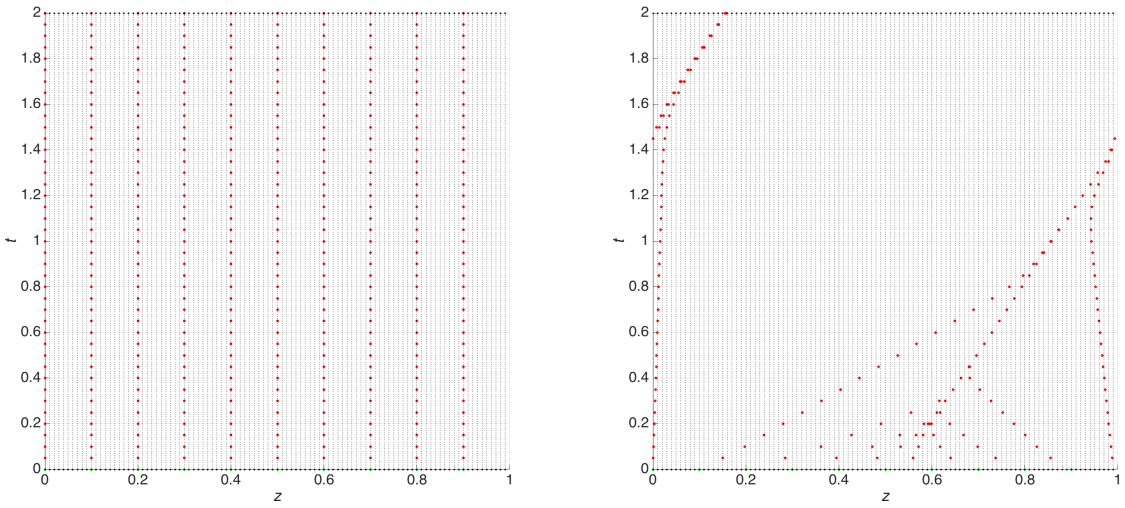
Figure 3.3: Solutions of Burgers' and Kuramoto-Sivasinsky equations. These solutions on a uniform mesh represent the truth from which to sample the observations. Their implementations on an adaptive moving mesh are used as forecast models of the ensemble.

a small area of the model domain; they can thus be more spatially localized compared to Eulerian observations. In our experiments with Lagrangian observations, if two observations come within the threshold distance, 10^{-3}, the one closer to the upper boundary of the spatial domain is omitted from the assimilation at that and all future observation times so as not to over-sample a specific location. As a result, the total number of Lagrangian observations will tend to decrease in time. An illustration of the different spatial coverage provided by the Eulerian and Lagrangian observations is given in Figure 3.4, for the BGM model with $d^{EUL} = d_0^{LAG} = 10$ on the mesh of the nature run.

3.6 Summary

In this chapter, we introduced the sea ice model neXtSIM as motivation for implementing data assimilation on an adaptive moving mesh model. The mesh is non-conservative due to the remeshing process, meaning it does not conserve the number of mesh points. We wanted to consider a one-dimensional analogue of the two-dimensional problem that preserved the essential features of neXtSIM, which we did in Section 3.3. We described the remeshing process implemented when mesh points become too close together or too far apart.

We also introduced periodic Burgers' and Kuramoto-Sivashinsky equations, denoted by BGM and KSM, respectively. We saw that with the initial conditions given, the solution of BGM consists



(a) Eulerian Observations

(b) Lagrangian Observations

Figure 3.4: Observations sampled from the truth (Fig. 3.3a) in Eulerian **(a)** and Lagrangian **(b)** sense mimicking geo-stationary satellite and buoy measurements, respectively.

of a moving front that dissipates, while KSM exhibits chaotic behavior. The former gives a simple model to test on, while the latter allows us to see whether our algorithm is robust.

CHAPTER 4

The First Approach to the Adaptive Mesh EnKF: Interpolation

4.1 The ensemble Kalman filter for an adaptive moving mesh model

We will introduce a modification of the EnKF [19] suitable for numerical models integrated on an adaptive moving mesh. The discussion herein pertains to the stochastic version of the EnKF [40], but the approach can be straightforwardly extended to deterministic EnKFs (see e.g. [41]) without major modifications. A recent review on EnKF-like methods and their applications to atmospheric circulation models can be found in [20]. The work here can also be found in [42].

The challenge of implementing an EnKF on adaptive moving mesh model with remeshing is that the dimension of the state vector will be potentiailly different for each ensemble member. This is addressed by [22] in which the idea of a fixed reference mesh, called the observation mesh, is introduced, which has higher resolution around the predefined observations. We will adopt this approach here but introduce a new variant in utilizing meshes of different resolutions. In particular, we will work with a high- and a low-resolution mesh. We see these as representing the extremes which should bracket the possible results of using meshes of various resolutions. They are, respectively associated with the two tolerance parameters δ_1 and δ_2, therefore linked directly to the mesh of the models while giving us the flexibility of assimilating any type of observations without prior information so as is generally the case in realistic applications. In addition, in our approach, the analyzed states are mapped back onto the adaptive moving meshes to preserve the mesh resolving fine scale structures generated by the dynamics of the models.

The location of the nodes and their total number are bound to change with time and across ensemble members: each member will now provide a distinct discrete representation of the underlying continuous physical process, based on a different number of differently located sample points. The individual ensemble members have to be intended now as as samples from a different partition of the physical system's state space and they do not provide a statistically consisten sampling of the discrete-in-space uncertainty distribution. This is reflected in practice by the fact that the members

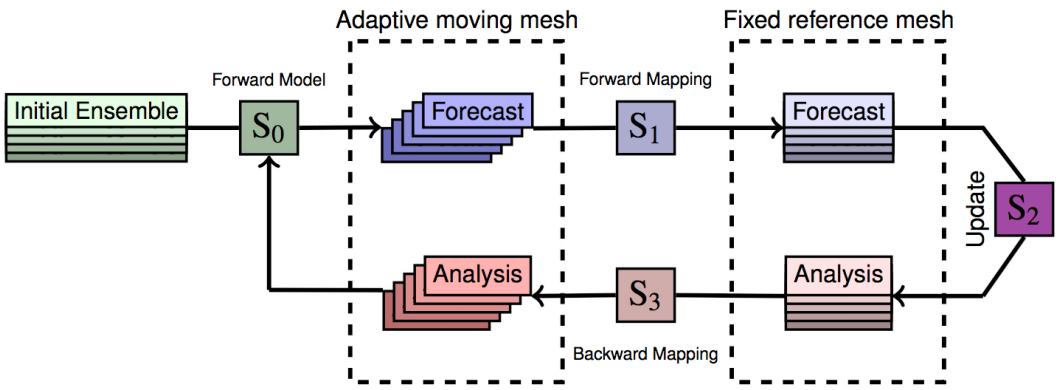


Figure 4.1: Illustration of the analysis cycle in the proposed EnKF method for adaptive moving mesh models. In S_1, adaptive meshes are mapped onto the fixed reference mesh. The ensemble is updated on the fixed reference mesh at step S_2 (i.e. the analysis). Then, in S_3, the updated ensemble members are mapped back to the corresponding adaptive moving meshes. The full process is illustrated in Figure 4.2 for one ensemble member. See text in Sect. 4.1 for full details on the individual process steps S_0, S_1, S_2, and S_3.

can no longer be stored column-wise to form ensemble matrices, and thus the matrix computations involved in the EnKF analysis to evaluate the ensemble-based mean and covariance cannot be performed.

On the other hand, on the reference mesh, on the reference mesh, the members are all samples from the same discrete distribution and can thus be used to compute the ensemble-based mean and covariance. The entire EnKF analysis process is carried out on this fixed reference mesh, and the results are then mapped back to the individual ensemble meshes. This procedure amounts to the addition of two steps on top of those in the standard EnKF. First, we map each ensemble member from its adaptive moving mesh to an appropriate fixed uniform mesh, and perform the analysis. Then, the updated ensemble members are mapped back to their adaptive moving meshes, providing the ensemble for the next forecast step.

The process is summarized schematically in Figureẽffig:flow. Steps S_0 and S_2, integration of the model $\mathcal{M}$ to compute prior statistics, and the analysis step, respectively, are common in an EnKF. At step S_1, before the analysis, the forecast ensemble on adaptive moving meshes is mapped onto the fixed uniform mesh, while step S_3 amounts for the back mapping from the fixed to the individual adaptive meshes. In the following sections, we give the details of processes in S_1, S_2, and S_3 following their respective order in the whole DA cycle.

4.1.1 Fixed reference meshes

We divide the physical domain $[0, L)$ into M cells of equal length, $\Delta\gamma$:

$$[0, L) = L_1 \cup L_2 \cup \cdots \cup L_M, \tag{4.1}$$

where $L_i = [\gamma_i, \gamma_{i+1})$. It follows that $\gamma_1 = 0$ and $\gamma_i = (i-1)\Delta\gamma$ for each i, and that $\gamma_{M+1} = L$. Because of the periodicity, we identify 0 and L in the fixed mesh; in other words, $\gamma_{M+1} = \gamma_1$ modulo L. The points γ_i are the *mesh nodes* of the fixed reference mesh.

While we are, in principle, free to choose the fixed reference mesh arbitrarily, it makes sense to tailor it to the application under consideration. We choose to define the resolution of this fixed uniform mesh based on the maximum and minimum possible resolution of the individual adaptive moving meshes in the ensemble. The resolution range in the adaptive moving mesh reflects the computational constraints adapted to the specific physical problem: it therefore behooves us to bring these constraints into the definition of the fixed mesh for the analysis.

The *high resolution* fixed reference mesh (HR) will be obtained by setting $M = N_1$ and the *low resolution* fixed reference mesh (LR) by setting $M = N_2$. We will focus on these two particular fixed meshes, although the methodology described below could be adapted to working with any fixed reference mesh. Recalling that $L = N_1\delta_1 = N_2\delta_2$, and the criteria for a valid mesh is given by (3.4), it can be seen that any valid mesh $\{z_1, z_2, \ldots, z_N\}$ will have at most one node in each cell L_i of an HR, and at least one node in each cell of an LR. In this chapter, we take the approach of using interpolation to fill in empty cells in the HR case; in the next chapter, we will use an alternative method. The LR case will average physical values at nodes that share a cell. It may seem that the higher resolution mesh would always be preferable, but a key finding of this work is that this is not always true.

Note that they hypothesis $L = N_1\delta_1 = N_2\delta_2$, i.e. the tolerances δ_1 and δ_2 are divisors of the domain length L, does not need to be assumed. The computational/physical constraints of the model may suggest δ_1 and δ_2 not satisfying this condition; it would be a technical change in our method to accommodate such a situation.

4.1.2 Mapping onto a fixed reference mesh

The mapping will take a state vector $\mathbf{x} = (u_1, u_2, \ldots, u_N, z_1, z_2, \ldots, z_N)$, where $\{z_1, z_2, \ldots, z_N\}$ is a valid mesh, onto a vector in $\mathbb{X}_M = \mathbb{R}^M \times V_M$ with $M = N_1$ (HR) or $M = N_2$ (LR). The state vector to which the map is applied should be thought of as an ensemble member at the forecast step, so that it has gone through remeshing after its final model evolution step. Thus, N may be any integer between N_1 and N_2. This is Step S_1 in the scheme of Figure 4.1.

We denote the mapping as $\mathcal{P}_j : \mathbb{X} \to \mathbb{X}_M$, with $M = N_1$ for $j = 1$ (HR) or $M = N_2$ for $j = 2$ (LR) as above. Recalling that the γ_i are nodes of the fixed reference mesh, the image of a specific $\mathbf{x} \in \mathbb{X}_N$ has the form

$$\mathcal{P}_j(\mathbf{x}) = (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_M, \gamma_1, \gamma_2, \ldots, \gamma_M). \tag{4.2}$$

The physical value $\tilde{u}_i$ is viewed as the value of u at mesh node γ_i the tilde is used hereafter to refer to quantities on the fixed reference mesh.

To set the u-values, we introduce a shifted mesh as follows: set $\tilde{L}_i = [\gamma_i - \delta/2, \gamma_i + \delta/2)$ for $i = 2, \ldots, M$ where $\delta = \delta_1$ or $\delta = \delta_2$ and again $M = N_1$ or $M = N_2$, respectively. Further, set $\tilde{L}_1 = [0, \delta/2) \cup [L - \delta/2, L)$. We view $\tilde{L}_1$ as an interval since we identify 0 and L. The procedure is now separated into the high and low resolution cases.

Case 1 - HR: Taking $\mathbf{x} \in \mathbb{X}_N$ as above, if there exists $z_k \in \tilde{L}_i$, then set $\tilde{u}_i = u_k$. If there is not $z_k \in \tilde{L}_i$ but there exists $x_k < \gamma_i$, then find k such that $z_k < \gamma_i < z_{k+1}$ and set

$$\tilde{u}_i = \frac{u_k + u_{k+1}}{2}. \tag{4.3}$$

If there is no such z_k, then set

$$\tilde{u}_i = \frac{u_1 + u_{N_1}}{2}. \tag{4.4}$$

underlineCase 2 - LR: For each i find all k such that $z_k \in \tilde{L}_i$. Denote these by $z_{k_i}, \ldots, z_{k_i+n_i}$. Then set

$$\tilde{u}_i = \frac{1}{n_i} \sum_{j=k_i}^{k_i+n_i} u_j. \tag{4.5}$$

The map $\mathcal{P}_j$ is now well-defined, in both the HR and LR cases, for each $\mathbf{x} \in \mathbb{X}_N$.

For the EnKF, we also require the map that omits the mesh points in the fixed reference mesh:

$$\tilde{\mathcal{P}}_j(\mathbf{x}) = (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_M), \tag{4.6}$$

where again $M = N_1$ or N_2 for HR or LR, respectively.

In the EnKF analysis, we will denote by $\tilde{\mathcal{P}}_j(\mathbf{x})$ by $\tilde{\mathbf{u}}$ and work with this reduced state vector, which consists only of the physical values and not the mesh points. A consequence is that we will not be updating the mesh locations, but rather re-mapping the analysis back onto the original adaptive mesh for each ensemble member. We will discuss the possibility of updating the mesh locations in the conclusions.

4.1.3 Observation operator

The observations will be of physical values ($\mathbf{u}$) at specific locations ($\mathbf{z}^o$). Assuming there are d observations, the observation operator will be a mapping on reduced state vectors $\tilde{\mathbf{u}} = (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_M)$ given as $\mathbf{y} = \mathcal{H}(\tilde{\mathbf{u}})$, i.e., $\mathcal{H} : \mathbb{R}^M \to \mathbb{R}^d$ with $M = N_1$ or N_2. Each component of $\mathcal{H}(\tilde{\mathbf{u}})$ is the estimate the state vector $\tilde{\mathbf{u}}$ gives of the observations at locations $\mathbf{z}^o$. For the explicit representation of the observation operator, let us consider one observation at once, so that for all $1 \leq j \leq d$ we consider the j-th observation and find i such that $z_j^o \in L_i$; then the j-th component of the observation operator reads:

$$h_j(\tilde{\mathbf{u}}) = \tilde{u}_i + \frac{z_j^o - \gamma_i}{\gamma_{i+1} - \gamma_i} \left(\tilde{u}_{i+1} - \tilde{u}_i\right). \tag{4.7}$$

Since $\gamma_i \leq z_j^o < \gamma_{i+1}$, this is the natural linear interpolation between the values of $\mathbf{u}$ at γ_i and γ_{i+1}. The full observation operator is then

$$\mathcal{H}(\tilde{\mathbf{u}}) = \left(h_1(\tilde{\mathbf{u}}), h_2(\tilde{\mathbf{u}}), \ldots, h_d(\tilde{\mathbf{u}})\right), \tag{4.8}$$

where each $h_j(\tilde{\mathbf{u}})$ has the above form of an observation value at their respective observation locations z_j^o.

Thus, we can eventually define the state vector on $\tilde{\mathbf{\Gamma}}$ as

$$\tilde{\mathbf{w}}(t) = \begin{pmatrix} \tilde{\mathbf{x}}(t) \\ \tilde{\mathbf{z}}(t) \end{pmatrix} = [\tilde{x}_1(t)\ \tilde{x}_2(t)\ \ldots\ \tilde{x}_{M-1}(t)\ \tilde{x}_M(t)\ \tilde{z}_1(t)\ \tilde{z}_2(t)\ \ldots\ \tilde{z}_{M-1}(t)\ \tilde{z}_M(t)]^{\mathrm{T}}. \tag{4.9}$$

4.1.4 Analysis using the ensemble Kalman filter

After mapping all the ensemble members onto the dedicated fixed reference mesh (either the high- or low-resolution one), the stochastic EnKF can be applied in the standard way. This is Step S_2 in our scheme. The mapped forecast ensemble members can be stored as columns of the forecast ensemble matrix

$$\mathbf{E}^f = \left[\tilde{\mathbf{u}}_1^f\ \ldots\ \tilde{\mathbf{u}}_{N^e}^f\right] \in \mathbb{R}^{M \times N^e}, \tag{4.10}$$

with $M = N_1$ or $M = N_2$ for HR and LR reference meshes, respectively, with N^e being the ensemble size. To simplify the notation, the time index and the tilde from the matrices are omitted: all terms entering the analysis update operations are defined at the same analysis time onto the fixed, either HR or LR, mesh. The forecast ensemble mean is computed as

$$\bar{\tilde{\mathbf{u}}}^f = \frac{1}{N^e} \sum_{n=1}^{N^e} \tilde{\mathbf{u}}_n^f, \tag{4.11}$$

while the normalized forecast anomaly matrix $\mathbf{X}^f$ is formed by subtracting the forecast ensemble mean from each of the ensemble members as

$$\mathbf{X}^f = \frac{1}{\sqrt{N^e - 1}} \left[\tilde{\mathbf{u}}_1^f - \bar{\tilde{\mathbf{u}}}^f\ \ldots\ \tilde{\mathbf{u}}_{N^e}^f - \bar{\tilde{\mathbf{u}}}^f\right]. \tag{4.12}$$

Model outputs are confronted with the observations at the end of every analysis interval, and are stored in the observation vector, $\mathbf{y} \in \mathbb{R}^d$. The observations are related to the system state via the (generally nonlinear) observational model

$$\mathbf{y} = \mathcal{H}(\tilde{\mathbf{u}}) + \epsilon \tag{4.13}$$

and are assumed affected by a Gaussian, zero-mean white-in-time noise ϵ with covariance $\mathbf{R} \in \mathbb{R}^{d \times d}$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. In the experiments described later in this chapter, we directly observe the model

physical variables (onto the fixed reference mesh), $\tilde{\mathbf{u}}$, so that the observation operator only involves a linear interpolation, and is thus linear. Nevertheless, the approach herein described is suitable to work with nonlinear $\mathcal{H}$ subject to minor modifications.

In the stochastic EnKF [40], the observations are treated as random variables, so that each ensemble member assimilates a different perturbed observations vector

$$\mathbf{y}_n = \mathbf{y} + \epsilon_n, \quad 1 \leq n \leq N^e, \tag{4.14}$$

with $\epsilon_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. We can now compute the normalized anomaly ensemble of the observations

$$\begin{aligned} \mathbf{Y}_o &= \frac{1}{\sqrt{N^e - 1}} \left[\mathbf{y}_1 - \mathbf{y} \ \ldots \ \mathbf{Y}_{N^e} - \mathbf{y} \right] \\ &= \frac{1}{\sqrt{N^e - 1}} \left[\epsilon_1 \ \ldots \ \epsilon_{N^e} \right], \end{aligned} \tag{4.15}$$

and define the ensemble-based observational error covariance matrix,

$$\mathbf{R}^e = \mathbf{Y}_o \left(\mathbf{Y}_o \right)^{\mathrm{T}}, \tag{4.16}$$

and the observed ensemble-anomaly matrix,

$$\mathbf{Y} := \mathcal{H}\left(\mathbf{E}^f \right) - \mathcal{H}\left(\bar{\mathbf{E}}^f \right), \tag{4.17}$$

with $\bar{\mathbf{E}}^f = \bar{\tilde{\mathbf{u}}}^f \mathbf{1}$ and $\mathbf{1} = [1 \ \ldots \ 1]^{\mathrm{T}} \in \mathbb{R}^M$. The forecast ensemble members are then individually updated according to

$$\tilde{\mathbf{u}}_n^a = \tilde{\mathbf{u}}_n^f + \mathbf{K} \left[\mathbf{y}_n - \mathcal{H}\left(\tilde{\mathbf{u}}_n^f \right) \right], \quad 1 \leq n \leq N^e, \tag{4.18}$$

where

$$\mathbf{K} = \mathbf{X}^f \mathbf{Y}^{\mathrm{T}} \left[\frac{1}{N^e - 1} \mathbf{Y}\mathbf{Y}^{\mathrm{T}} + \mathbf{R}^e \right]^{-1} \tag{4.19}$$

is the ensemble-based Kalman gain matrix. It is worth recalling that in the limit, $N^e \to \infty$, $\mathbf{R}^e \to \mathbf{R}$ and the Kalman gain, $\mathbf{K}$, converges to that of a classical, full rank Kalman filter if both the dynamical and the observational models are linear and all of the errors are Gaussian.

When applied to large dimensional systems, for which $N^e \ll M$ as typical in the geosciences,

the success of the EnKF is related to the use of localization and inflation (see e.g. Section 4.4 of [17] for a review). In this work localization is not used, but the covariance multiplicative scalar inflation ([43]) is adopted, so that the ensemble-based forecast anomaly matrix is inflated as

$$\mathbf{X}^f \mapsto \alpha \mathbf{X}^f \tag{4.20}$$

with $\alpha \geq 1$, before $\mathbf{X}^f$ is used in the analysis update (4.18). Multivariate multiplicative inflation or more sophisticated adaptive inflation procedures exist and could have been implemented here, but this is not of great importance in this work, and the scalar coefficient α has been properly tuned. A recent review of adaptive inflation methods can be found in [44].

The update analysis ensemble in (4.18) is then used to initialize the next forecast step. However, prior to this, we need to map back each individual analysis member on their respective adaptive, non-uniform, mesh; the process is described in the next section.

4.1.5 From a fixed reference mesh to an adaptive moving mesh

After the analysis, the update on the fixed reference mesh has to be mapped back on the individual adaptive moving meshes of the ensemble members. In the forward mapping step S_1 (see Figure 4.1), the mapping indices associating the nodes in the adaptive moving mesh with nodes in the reference mesh, are stored in an array. These are the indices resulting from the projections on the HR or the LR reference mesh as previously described.

Each analysis ensemble member $\tilde{\mathbf{u}}_n^a$ will thus retrieve its adaptive mesh $z_1, z_2, \ldots, z_{N(n)}$ from the stored array. In the reverse mapping step S_3 (Figure 4.1), the updated solution is projected to the adaptive moving meshes by locating each z_j in an interval $\tilde{L}_m$ and assigning the m-th component of $\tilde{\mathbf{u}}_n^a$ to be the i-th component of u in the vector $\mathbf{x}_k$ that will initialize the model after the analysis time step.

In summary, each ensemble member after the analysis step will have the form

$$\mathbf{x}_k = (u_1 \; u_2 \; \ldots \; u_N \; z_1 \; z_2 \; \ldots \; z_N) \tag{4.21}$$

where if $z_i \in \tilde{L}_m$, then $u_i = \tilde{\mathbf{u}}(\gamma_m)$. The backward mapping procedure is the same for both HR and LR cases, although it will provide different results.

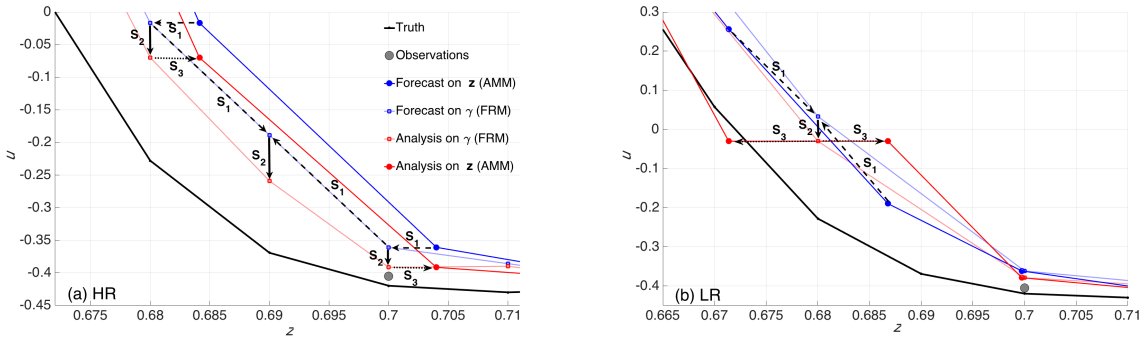


Figure 4.2: Schematic illustration of the DA cycle on the high resolution **(a)** and low resolution **(b)** fixed reference mesh where only **u** is updated. AMM and FRM stands for adaptive moving mesh and fixed reference mesh, respectively. Dark and pale blue/red lines are forecast/analysis on adaptive moving mesh and fixed reference mesh, respectively. Gray circles are the observations. Following the arrows: S_1 is the mapping the adaptive moving mesh on to the fixed reference mesh, S_2 is the update of the ensemble member, S_3 is the backward mapping on the adaptive moving mesh (see Fig. 4.1).

The process steps $S_1 \rightarrow S_2 \rightarrow S_3$ are illustrated in Figure 4.2, representing HR/LR cases, respectively, for one ensemble member, and using the Burgers' equation as a model ([1]); the experimental setup is described in the next section.

Let us first consider the HR case of Figure 4.2 (a). In S_1, the forecasted physical quantity $\mathbf{u}^f$ on the adaptive moving mesh (dark blue with large circles) is mapped to the fixed reference mesh nodes (light blue with small circles) at $\gamma_{m-1} = 0.68$ and$\gamma_{m+1} = 0.70$. The fixed mesh's node $\gamma_m = 0.69$ is left empty; thus, a value is assigned by interpolation from the adjacent nodes γ_{m-1} and γ_{m+1}. This provides the forecast physical quantity, $\tilde{\mathbf{u}}^f$, on the full reference mesh and completes step S_1. In the next step, S_2, $\tilde{\mathbf{u}}^f$ is updated using the stochastic EnKF as described in the previous section to get the analysis field $\tilde{\mathbf{u}}^a$ (light red line and small circles). Finally, in step S_3, $\tilde{\mathbf{u}}^a$ is mapped back to the adaptive moving mesh so as to get $\mathbf{u}^a$ (dark red line with large circles). We note that the physical quantity on the interpolated node γ_m in the fixed reference mesh is not mapped back (as that node "did not exist" in the original adaptive mesh), yet it was required during step S_2 to perform the analysis.

Similarly, Figureẽffig:aflow (b) describes the LR case. In this situation, however, the forecasted physical quantity on the adaptive moving mesh nodes at 0.672 and 0.686 are averages (step S_1) in order to associate a vlue on the fixed reference mesh node $\gamma_m = 0.68$ before the analysis. After the update (step S_2), in step S_3 the analysis $\tilde{u}_m$ at $\gamma_m = 0.68$ is used to provide the analyses on both of

the original nodes (at 0.672 and 0.686) on the adaptive mesh; this means they will have the same analyzed value. As a result, we observe that the anlysis is better than the forecast (in the sense of being closer to the truth: compare dark blue/red circles, respectively, for forecast and analyses) at node z =0.672 but worse at node z =0.686. In the latter case, in fact, the overestimate of the truth passes from about 0.15 to more that 0.3 for the forecast and analysis, respectively. On the other hand, at node z =0.672 the forecasted overestimate of about 0.2 is reduced to a slight understimate of about 0.04.

We make a remark on a key aspect of our methodological choice: the ratio of the remeshing criteria $\frac{\delta_2}{\delta_1}$ exerts a control on the relation between the adaptive moving meshes and the fixed reference mesh. In fact, $\frac{\delta_2}{\delta_1}$ is the upper bound of the number of nodes that will be interpolated in the HR case, and averaged in the LR case respectively, since it represents the maximum ratio between the dimension of the fixed reference mesh and that a moving mesh can never reach.

4.2 Experimental setup

In the experiments that follow, we have chosen to deploy as many Lagrangian observers at t_0 as Eulerian ones and to place them at the same locations, i.e. $d_0 = d$. The number of Eulerian observations, and the initial number of Lagrangian observations, is set to $d^{EUL} = d_0^{LAG} = 10$ and $d^{EUL} = d_0^{LAG} = 20$ for BGM and KSM, respectively. Gaussian, white-in-time, spatially uncorrelated noise is added to these observations; the observational error covariance matrix is diagonal, so that $\mathbf{R} = \sigma_o^2\mathbf{I}$, with σ_o being the observational error standard deviation and $\mathbf{I}$ the identity matrix. These synthetic observations are assimilated with the modified EnKF we presented, and the specifications of its implementation, namely the number of initial ensemble members, initial mesh size, and inflation, are provided in Section. The analysis interval is set to $\Delta t = 0.05$ time units in all the DA experiments and for both models and observation types. A summary of the experimental setup is given in Table 4.1.

The experiments are compared by looking at the root mean square error (RMSE) of the ensemble mean (with respect to the natural run) and the ensemble spread. Since the analysis is performed on either the HR or the LR fixed mesh, the computation of the RMSE and spread is done on the mesh resulting from their intersection. Given that we have chosen the remeshing criteria in both models such that δ_1 is half of δ_2, the intersection mesh is the LR mesh itself. Finally, in all of the experiments, the time-mean of the RMSE and spread are computed after $T = 1$ time units, unless

Table 4.1: Experimental setup parameters: ν is the viscosity, δ_1 and δ_2 are the remeshing criteria, N_1 and N_2 the number of nodes in the HR and LR fixed reference mesh, T the duration of the experiments, Δt the analysis interval, d^{EUL} and d_0^{LAG} the number of Eulerian observations and the initial number of Lagrangian observations.

Model	ν	δ_1/δ_2	N_1/N_2	T	Δt	d/d_0	σ_0
BGM	0.008	0.01/0.02	100/50	2	0.05	10	0.01
KSM	0.027	$0.02\pi/0.04\pi$	100/50	5	0.05	20	0.78

stated otherwise.

4.3 Results

We present the results in three subsections. In the first two subsections, we investigate the modified EnKF with fixed reference mesh (either HR or LR), for the BGM and KSM, respectively, using Eulerian observations. In these sections, we also present the tuning of the EnKF with respect to the ensemble size (N^e), inflation factor (α), and initial adaptive moving mesh size (N_0). The combination of these parameters giving the best performance with BGM is then kept and used in the next section, where the comparison between Eulerian and Lagrangian observation cases is described.

4.3.1 Modified EnKF for adaptive moving mesh models - Burgers' equation

In this section, the experiments using BGM are presented. In order to calculate the base error due to the choice of the specific fixed reference mesh, HR or LR, and the resulting mapping procedures, we first perform and ensemble run without assimilation. This DA-free ensemble run is subject to all of the steps described in Figure 4.1 except for S_2, in which the analysis update is performed. Given that DA is not carried out, the difference between the HR and LR experiments (if any) can only be due to the mapping procedure. Recall that this procedure differs in that it involves interpolation or averaging in the HR or LR cases, respectively. For consistency, the mapping to/from the fixed reference mesh is performed every $\Delta t = t_{k+1} - t_k$, i.e. the time between the assimilation of observations.

Figure 4.3 displays the RMSE and the ensemble spread for the HR and LR in these DA-free ensemble runs. We see that the RMSE is slightly larger in the LR than in the HR case, indicating that averaging introduces larger errors than interpolation in this specific model scenario. This can be interpreted in terms of the sharpness of the Burgers' solution that might not be accurately

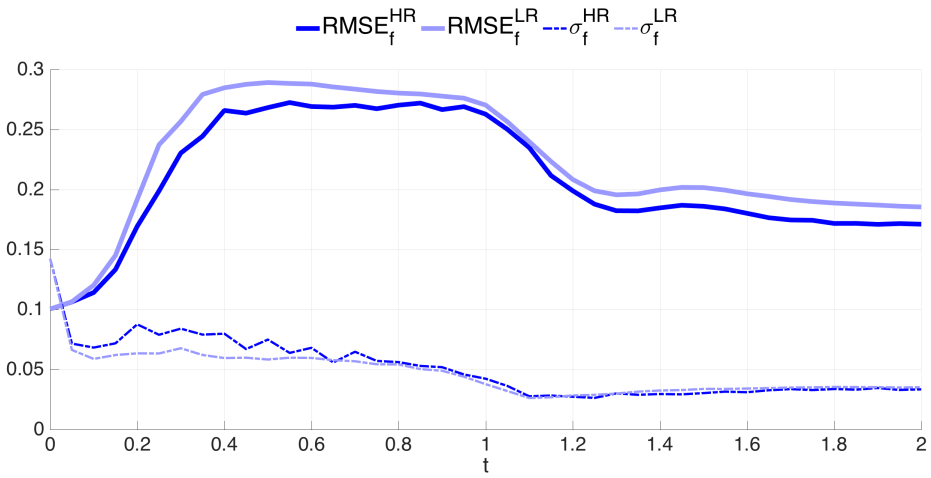


Figure 4.3: Time evolution of the forecast RMSE (solid line) and the spread (σ, dashed line) of DA-free ensemble run using BGM. Dark and light lines represent the HR and LR, respectively.

described using the LR mesh. Furthermore, this is also consistent with the previous observation that the LR analysis was deteriorating the forecast in some instances. After an initial faster error growth in the LR case, at about $t = 0.4$, the difference between LR and HR almost stabilizes, with the two error curves having the same profile. The ensemble spread is initially slightly larger in the HR case, but then it attains similar values for both HR and LR after $t = 0.6$, suggesting that the spread does not depend critically on the type of mapping and resolutions of the fixed reference mesh. While this appears to be a reasonable basis for building the EnKF, Figure 4.3 also highlights the undesirable small spread of values compared with the RMSE. We will come back to this point in the DA experiments to follow.

In the DA experiments, we study the sensitivity of the EnKF to the ensemble size, inflation factor, and initial size of the adaptive moving meshes. Recall that the ensemble members are all given the same uniform mesh at the initial time; however, these meshes will then inevitably evolve into a different, generally non-uniform mesh for each member. We remark that the three parameters under consideration are all interdependent and a proper tuning would involve varying them all simultaneously, which would make the number of experiments grow too much. To reduce the computational burden, we have opted instead to vary only one at a time, while keeping the other two fixed.

The results of this tuning are displayed in Figure 4.4, showing the RMSE of the EnKF analysis

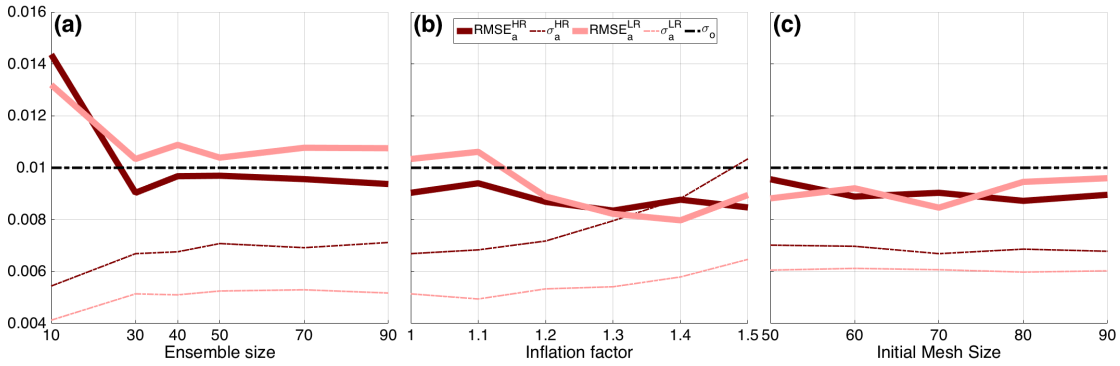


Figure 4.4: Time-mean of the RMSE of the analysis ensemble mean (solid line) and ensemble spread (σ, dashed line) of BGM for different ensemble size N^e **(a)**; inflation factor, α (**b**); and initial mesh size, N_0 **(c)**. Dark and light red show the HR and LR, respectively.

(the ensemble mean), and the spread, as a function of the ensemble size, inflation factor, and initial mesh size, respectively, in panels **a**, **b**, and **c**. The RMSE and spread are averaged in space and time, after the initial spin-up period $T = 1$. For reference, we have also plotted the observational error standard deviation (horizontal black dashed line).

In the case of the sensitivity to the ensemble size (Figure 4.4(a)), N^e is varied between 10 and 90, while the initial mesh size is kept to 70 for both HR and LR cases, and inflation is not used (i.e., $\alpha = 0$). The RMSE in the HR case is generally lower than in the LR case, which are respectively slightly below and above the observational error standard deviation. In both cases, however, the RMSE approximately converges to quasi-stationary values as soon as $N^e \geq 30$. This phenomenon, that we also observe for KSM in the next section, is reminiscent of the behavior in a chaotic system, where the EnKF error converges when N^e is larger or equal to the dimension of the unstable-neutral subspace of the dynamics ([]).

We therefore set $N^e = 30$ and study the sensitivity to the inflation factor in Figure 4.4(b) (the initial mesh size is still kept to 70). Inflation is expected to mitigate the difference (the underestimation) between the RMSE and spread shown in Figure 4.4(a). By looking at Figure 4.4(b), this actually seems to be the case and in the LR case, the RMSE decreases and the spread increases by increasing the inflation factor α. In the HR case, the RMSE is already lower than the observational standard deviation and the inflation has only a small effect; the increase of spread is not accompanied by a similar decrease of error. Based on this, we hereafter set the inflation to $\alpha = 1$ for HR and $\alpha = 1.45$ for LR.

Table 4.2: Ensemble size (N^e), inflation factor (α), and initial mesh size (N_0) chosen from the sensitivity experiments in Figure 4.4 to perform the experiment in Figure 4.5. Resulting time mean of the RMSE and spread (σ) for the HR and LR using BGM between $t = 0$ and 2 are also listed.

BGM	N^e	α	N_0	$RMSE_f$	$RMSE_a$	σ_f	σ_a
HR	30	1.0	70	0.025	0.023	0.026	0.015
LR	30	1.45	70	0.018	0.017	0.023	0.014

Finally, in Figure 4.4(c), we consider the initial mesh size; recall that the ensemble size is set to $N^e = 30$. Also recall that the size of the individual ensemble member's adaptive moving mesh size, N, is controlled by the remeshing tolerances $\delta_1 = 0.01$ and $\delta_2 = 0.02$, and can vary throughout the integration between 50 and 100. In the set of experiments depicted in Figure 4.4(c), we initialize the ensemble on an adaptive moving mesh of size N_0 ranging from 50 to 90. Interestingly, the EnKF does not exhibit great sensitivity to N_0 and the differences between HR and LR appear to be very small and not systematic. The fact that LR kept the the RMSE at the level of the HR case is the result of successful tuning. We saw, in fact, that the mapping error in the LR case is larger (cf Figure 4.3). Nevertheless, this initial disadvantage of the LR has been largely compensated by the inflation. In the experiments that follow, we have chosen to fix $N_0 = 70$ for both HR and LR.

The results of the tuning experiments in Figure 4.7 and selected value of the parameter are reported in Table 4.2, and are used in the experiments of Figure 4.5 that shows the forecast/analysis RMSE and spread for both HR and LR as a function of time. Notably, the HR and LR perform quite similarly for $t > 1.2$, when the solution of the model is possibly of small amplitude due to the viscous damping. Nevertheless, for $t \leq 1.2$, LR is often as good as ($t < 0.4$) or better ($0.4 \leq t \leq 1.2$) than HR, making LR a viable, computationally more economic, solution. The time-averaged RMSE and spread of these experiments are included in Table.

4.3.2 Modified EnKF for adaptive moving mesh models - Kuramoto-Sivashinsky equation

This section shows the same type of results as in the previous section, this time applied to the KSM. We begin by evaluating the errors relating to the mapping on the HR and LR case by running a DA-free ensemble; results are shown in Figure 4.6.

As opposed to what is observed in Figure 4.3, we see now that the different mapping procedures in the HR and LR cases induce similar errors and impact the spread in a similar way. This difference

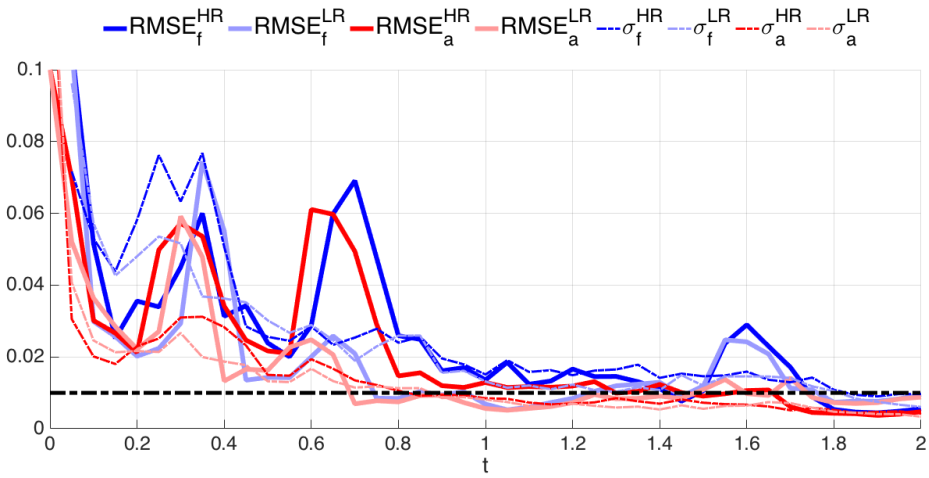


Figure 4.5: Time evolution of the RMSE (solid line) and spread (σ, dashed line) for BGM until $t = 2$. Dark and light lines represent the HR and LR, respectively. Blue and red show forecast and analysis, respectively.

is certainly due to the different dynamical behavior in BGM and KSM, with the solution of the latter displaying oscillations over all of the model domain. These can be, in some instances, well represented (i.e., less affected) by the averaging procedure in the LR case, in others by the interpolation in the HR case. Another remarkable difference with respect to BGM is that tnow the ratio spread/RMSE is larger, meaning that the spread is underestimating the RMSE relatively less than for BGM.

Figure 4.7 shows the same set of experiments as in Figure 4.4, this time using KSM. The time-mean of the RMSE and spread are again considered after $t = 1$, but experiments are run until $t = 5$ since KSM is not as dissipative as BGM with chosen values for the viscosity. Furthermore, all values are normalized using an estimate of the internal model variability based on the spin-up integration from $t = 0$ to $t = T = 20$.

In Figure 4.4(a), the analysis RMSE and spread are shown against the ensemble size, N^e. No inflation is applied and the initial mesh size is chosen to be 80 in both the HR and LR cases. The analysis RMSE passes below the observation error standard deviation as soon as $N^e = 30$ in the HR case, but an ensemble as large as $N^e = 50$ is required in the LR case. Based on these results, we have chosen to use $N^e = 40$ for both cases as a trade-off between computational cost and accuracy, given that the RMSE in the LR case is very close to observational accuracy. Notably, the spread is quite large in both cases, even larger than the RMSE in the HR configuration. With $N^e = 40$, the impact of inflation is considered in Figure 4.4(b). We see here how the spread is consistently

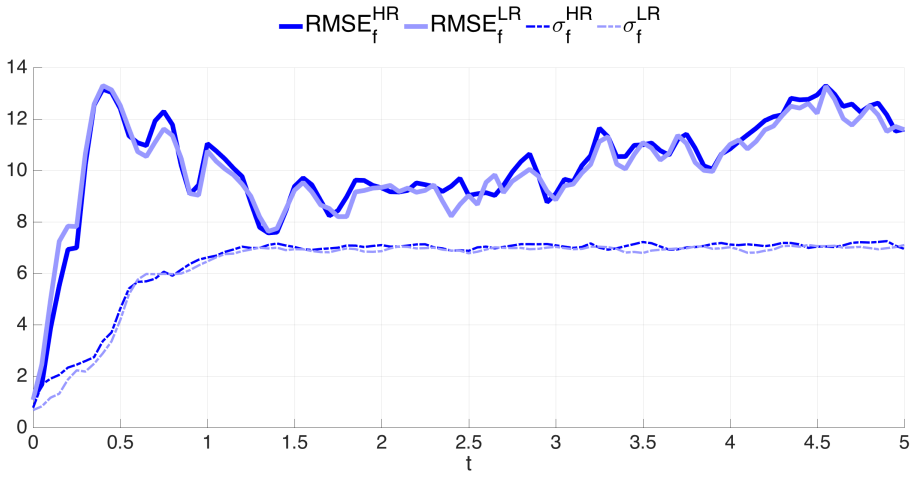


Figure 4.6: Same as Figure 4.3 but using KSM.

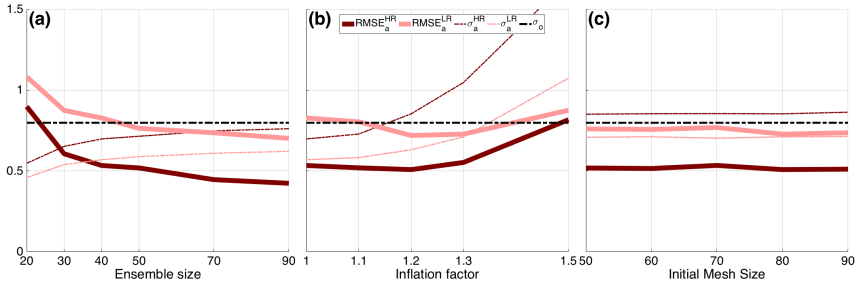


Figure 4.7: Same as Figure 4.4 but using KSM.

Table 4.3: Same as Table 4.2 but using KSM deduced from experiments in Figure 4.7.

KSM	N^e	α	N_0	$RMSE_f$	$RMSE_a$	σ_f	σ_a
HR	40	1.2	80	1.30	0.51	2.96	0.85
LR	40	1.3	80	1.25	0.78	1.83	0.71

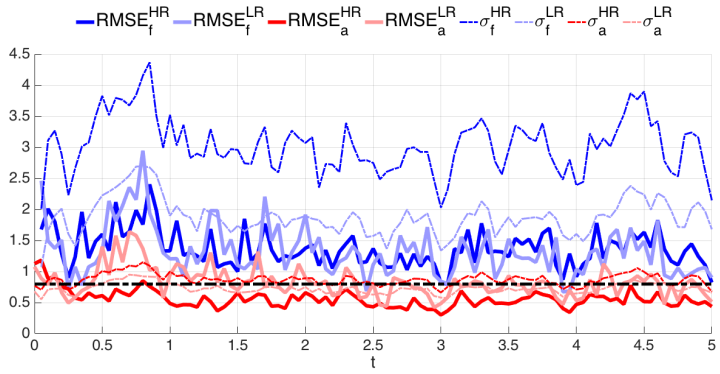


Figure 4.8: Same as Figure 4.5 but using KSM.

increased by increasing the inflation factor α and the corresponding RMSEs decrease until $\alpha = 1.3$ and increase afterward, possibly as a consequence of too much spread. The selected values for the inflation factor are $\alpha = 1.2$ and 1.3 for the HR and LR cases, respectively. Figure 4.7(c) studies the sensitivity to the initial mesh size, N_0. Similarly as to what is observed for the BGM in Figure 4.4(c), the performance of the EnKF does not show a marked sensitivity to N_0: it is arguable that the mesh size of the individual members quickly adjust to the values with little memory of the initial dimension. In the experiments that follow, the initial mesh size is set to $N_0 = 80$ in both HR and LR configurations. Overall, Figure 4.7 indicates that, as opposed to BGM, with KSM the EnKF on the HR reference mesh is always superior to the LR fixed mesh. The selected optimal values of N^e, α, and N_0 are reported in Table 4.3.

Figure 4.8 shows the time evolution of the forecast and analysis RMSE and spread for HR and LR until $t = 5$ using these selected values. First, we observe that the analysis RMSE is always lower than the corresponding RMSE of the forecast in both the HR and LR cases. Remarkably, the spread of the forecast is also larger than the RMSE of the forecast, in both configurations, indicating healthy performance of the EnKF. As for the comparison between HR and LR, we see that now the former is systematically better than the latter, suggesting that in the KSM, the benefit of performing

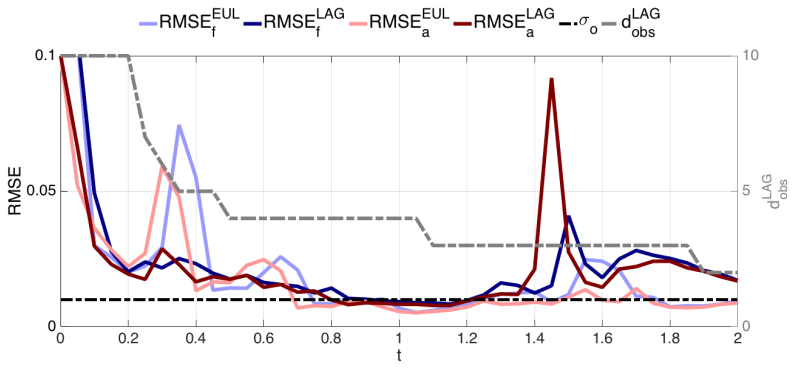


Figure 4.9

the analysis on HR are larger compared to BGM. Nevertheless, the LR case also performs well, and it could well be preferred when computational constraints are taken into consideration. The time-averaged RMSEs and spreads are reported in Table.

4.3.3 Impact of observation type: Eulerian versus Lagrangian

Up to this point, we have solely utilized Eulerian observations. Using the optimal setup presented in the previous section, we now assess the impact of different observations types, i.e. Eulerian or Lagrangian (see Figure 3.4(a) and Figure 3.4(b)). We consider there only the BGM with the LR configuration for the fixed reference mesh and the values for the experimental parameters are those in Table (first three columns of the second row). Results (not shown) with the KSM using Lagrangian observations indicate that the EnKF was not able to track the true signal, possibly as a consequence of the Lagrangian observers ending trapped within only few of the many fronts of the KSM solution (see Figure 3.3(b)): the number of observations and their distribution then becomes insufficient.

Figure 4.9 shows the forecast and analysis RMSE as a function of time, for both Eulerian and Lagrangian data. As for previous figures, the observation error standard deviation is superimposed as a reference, but the number of Lagrangian observers is now included (right y-axis). Recall that Lagrangian data are bound to decrease with time (cf. Section and Figure b) and that their initial number and locations are the same as for the Eulerian observations, i.e., $d^{EUL} = d_0^{LAG} = 10$ and they are equally spaced.

At first sight, one can infer from Figure 4.9 that overall Lagrangian data are approximately as effective as their Eulerian counterparts, even though they are fewer in number. This is reminiscent

of a known advantage of Lagrangian observations that has been documented in a number of studies (see, e.g., [], [], [], [], and references therein); although the actual positions at which the observations are made are assimilated in these pieces of work. A closer inspection of Figure 4.9 reveals also other aspects. For instance, it is remarkable that in the time interval $0.2 \leq t \leq 0.4$, the assimilation of $5 \leq d_{obs}^{LAG} \leq 10 = d_{obs}^{EUL}$ Lagrangian observations is superior to using $d_{obs}^{EUL} = 10$ fixed, evenly distributed Eulerian ones. On the other hand, when $t \geq 1.3$, the assimilation of Eulerian data is always bettern than Lagrangian, a behavior possibly due to the fact that $d_{obs}^{LAG} \leq 3$ and that, despite their dynamically guided locations, they are not as many as required to keep the error low. It is finally worth pointing out that the episode of very high analysis RMSE (higher than the corresponding forecast RMSE) occurring at $t = 1.5$; the assimilation in that case was very clearly detrimental. Nevertheless, the EnKF was able to recover quickly and the RMSE was reduced to much smaller values, close to the observation error.

4.4 Conclusions

In this chapter, we proposed a novel methodology to perform ensemble data assimilation with computational models that use a non-conservative adaptive moving mesh. Meshes of this sort are said to be adaptive because their node locations adjust to some prescribed rule that is intended to improve model accuracy. We have focused here on models with a Lagrangian solver, in which the nodes move following the model's velocity field. They are said to be non-conservative because the total number of nodes in the mesh can itself change when the mesh is subject to remeshing. We have considered the case in which remeshing avoids having nodes too close or too far apart than given tolerance distances; in practice the tolerances define the set of valid meshes. When an invalid mesh occurs throughout the integration, it is then remeshed and a valid one is created.

The major challenge for ensemble data assimilation stands in that the dimensions of the state space changes in time and differs across ensemble members, impeding the normal ensemble-based operations (i.e, matrix computations) at the analysis update. To overcome this issue, we have added in our methodology one forward and one backward mapping step before and after the analysis, respectively. This mapping takes all the ensemble members onto a fixed, uniform reference mesh. On this mesh, all ensemble members have the same dimension and are defined on the same spatial mesh, thus the assimilation of data can be performed using standard EnKF approaches. We have used the stochastic EnKF, but the approach can be easily adapted to the use of a square-root EnKF.

After the analysis, the backward mapping returns the updated values to the individual, generally different, and non-uniform meshes of the respective ensemble members.

We consider two cases: a high resolution and a low resolution fixed uniform reference mesh. The essential property is that their resolution is determined by the remeshing tolerances δ_1 and δ_2, such that the high- and low-resolution fixed reference meshes are the uniform meshes that bound, from above and below respectively, the resolution of all relevant adaptive meshes. While one can in principle use a fixed reference mesh of arbitrary resolution, our choice connects the resolution of the reference mesh to the given physical and computaitonal constraints, reflected by the tolerance values in the model design. This in practice means that our reference mesh cell will contain at most, or at least, one node of the ensemble member mesh, in either the high- or low-resolution cases respectively. Hence, using this characterization, we can avoid excessive smoothing or interpolation at the mapping stages. Depending on whether the tolerances are divisors of the model domain dimension, the reference meshes can also be themselves valid meshes; nevertheless, this condition is not required for the applicability of our approach.

We tested our modified EnKF using two 1D models, the Burgers and Kuramoto-Sivashinsky equations. A set of sensitivity tests are carried through some key model and DA setup parameters: the ensemble size, inflation factor, and initial mesh size. We have considered two types of observations: Eulerian and Lagrangian. We have shown that, in general, a high resolution fixed reference mesh improves the estimate more than a low resolution fixed reference mesh. Whereas this might indeed by expeted, our results also show that a low-resolution reference mesh affords a very high level of accuracy if the EnKF is properly tuned for the context. The use of a low-resolution fixed mesh has the obvious advantage of a lower computational burder, given that the size of the matrix operations to be implemented at the analysis step scales with the size of the fixed reference mesh.

We then examined the impact of assimilating Lagrangian observations compared with Eulerian ones and have seen, in the context of Burgers' equation, that the former improves the situation as much as the latter. The effectiveness of Lagrangian observers, despite being fewer in number that for the case of fixed, Eulerian observations, comes from their concentrating where their information is most useful, i.e., withing the sharp single (shock-like) front of the Burgers solution.

In this chapter, we have focused on the design of the strategy and, for the sake of clarity, have focused only on updating the physical quantities, while the locations of the ensemble mesh nodes

were left unchanged. A natural extension of this is to subject both the model physical variables and the mesh locations to the assimilation of data. Both would then be updated at this time, and this is currently under investigation.

CHAPTER 5

The Second Approach to the Adaptive Mesh EnKF: No Interpolation

5.1 Introduction

As seen in the previous chapter, one of the key difficulties in developing an ensemble-based data assimilation method for use in an adaptive moving mesh was that the ensemble meshes were quite different. The mesh points, in general, are not at the same location for each ensemble member. Further, the number of mesh points differed for each ensemble member. To rectify this, each ensemble member was projected onto a reference mesh, for which there were two cases: high-resolution (HR) and low-resolution (LR). In the HR case, each cell was guaranteed at most one mesh point for each ensemble member; in the LR case, each cell contained at least one mesh point of each ensemble member. In the HR case, for a given ensemble member, cells that did not contain mesh points were filled in by interpolating from adjacent cells containing mesh points. In the LR case, cells containing more than one mesh point had their values averaged. This resulted in each cell containing exactly one mesh point [42], corresponding to exactly one physical value, in both the HR and LR cases. The interpolation and averaging justified using the exact ensemble size N^e when computing the ensemble statistics in the formulation of the EnKF.

This previous method, while successful, somewhat avoids the issue that not exactly N^e ensemble members are present in each cell. If there are fewer than N^e members present, as is possible in the HR case, mesh points are created based on information in surrounding cells; we are making a key assumption about the behavior of the ensemble member where we do not have a mesh point. Generally, the HR reference mesh is of sufficiently high resolution, and the ensemble members are well-behaved enough, that this assumption is not unreasonable. However, it is an assumption nonetheless, and it is possible that it could present an issue in more complicated models like Kuramoto-Sivashinsky (KSM).

We develop an alternative approach, applicable to the HR case, in this chapter that avoids the discussed interpolation. Cells without mesh points are left empty, and only the present ensemble

members contribute to the ensemble statistics. This requires a modification to the formulation of the version of the EnKF given in the previous chapter. In particular, the quantity N^e must be allowed to vary among cells. We give the formulation of this method here. We then apply this method to Burgers' equation (BGM). We run experiments with the same parameters as before, so as to make possible a direct comparison between the method with interpolation and the method without.

5.2 The EnKF for an adaptive moving mesh model - second version

5.2.1 The fixed reference mesh

As before, we divide the physical domain $[0, L)$ into M cells of equal length, $\Delta\gamma$:

$$[0, L) = L_1 \cup L_2 \cup \cdots \cup L_M, \tag{5.1}$$

where $L_i = [\gamma_i, \gamma_{i+1})$. We will assume periodic boundary conditions, so we identify 0 and L as before. For this formulation, we will only be working on the high resolution fixed reference mesh (HR). Thus, we will assume $M = N_1$ and $L = N_1\delta_1$.

5.2.2 Projecting onto the high-resolution fixed reference mesh

We use the word "projecting" here, as opposed to "mapping" in the previous chapter, to indicate the difference between the processes. We begin with a state vector $\mathbf{x} = (u_1, u_2, \ldots, u_N, z_1, z_2, \ldots, z_N)$, where $\{z_1, z_2, \ldots, z_n\}$ is a valid mesh, and we wish to embed it in a state vector $\mathbb{X}_M = \mathbb{R}^M \times V_M$, with $M = N_1$ since we are on the high-resolution reference mesh. As before, N can be any integer between N_1 and N_2.

We will denote the projection by $\mathcal{P}'_j : \mathbb{X} \to \mathbb{X}_M$, with $M = N_1$, recalling that the γ_i are nodes of the fixed reference mesh. Then the image of a specific $\mathbf{x} \in \mathbb{X}_M$ has the form

$$\mathcal{P}'_j(\mathbf{x}) = (\tilde{u}'_1, \tilde{u}'_2, \ldots, \tilde{u}'_M, \gamma_1, \gamma_2, \ldots, \gamma_M). \tag{5.2}$$

The physical value $\tilde{u}'_i$ should be thought of as the value of the physical variable u at the mesh mode γ_i, if the mesh point is present in the corresponding cell L_i.

Setting the u-values is a fundamentally different process here. We again introduce the shifted mesh $\tilde{L}_i = [\gamma_i - \delta_1/2, \gamma_i + \delta_1/2)$ for $i = 2, \ldots, N_1$; set $\tilde{L}_1 = [0, \delta_1/2) \cup (L - \delta_1/2, L)$, where we again

view $\tilde{L}_1$ as an interval due to periodicity. Taking $\mathbf{x} \in \mathbb{X}_{N_1}$ as above, if there exists $z_k \in \tilde{L}_i$, then set $\tilde{u}'_i = u_k$, as in the previous method. However, if there is no $z_k \in \tilde{L}_i$, then we do not set $\tilde{u}'_i$ to any value; we leave the corresponding entry vacant.

5.2.3 Observation operator

As in the previous chapter, the observations will be of physical values ($\mathbf{u}$) at specific locations ($\mathbf{z}^o$). We again assume there are d observations, so the observation operator will be a mapping $\mathcal{H} : \mathbb{R}^{N_1} \to \mathbb{R}^d$ Let $\tilde{\mathbf{u}}'$ be the state vector embedded in $\mathbb{R}^{N_1}$ with vacant entries, so that $\mathbf{y} = \mathcal{H}(\tilde{\mathbf{u}}') \in \mathbb{R}^d$. We cannot, however, use the same explicit representation as before, due to the vacant entries in the state vector.

Let us consider one observation, so that for all $1 \leq j \leq d$ we consider the j-th observation and find i such that $z_j^o \in L_i$. Let $i_1 = i$ if cell L_i is non-vacant; otherwise, let i_1 be the index of the nearest nonvacant cell to the left of cell L_i. Similarly, let $i_2 = i + 1$ if cell L_{i+1} is nonvacant; otherwise, let i_2 be the index of the nearest nonvacant cell to the right of cell L_i. Then the j-th component of the observation operator reads:

$$h_j(\tilde{\mathbf{u}}') = \tilde{u}'_{i_1} + \frac{z_j^o - \gamma_{i_1}}{\gamma_{i_2} - \gamma_{i_1}} (\tilde{u}'_{i_2} - \tilde{u}'_{i_1}). \tag{5.3}$$

5.2.4 Analysis using the ensemble Kalman filter

After projecting all the ensemble members onto the high-resolution fixed reference mesh, we can apply a modified version of the stochastic EnKF. The projected forecast ensemble members can be stored as columns of the forecast ensemble matrix

$$\mathbf{E}^{f'} = \left[\tilde{\mathbf{u}}_1^{f'} \ \dots \ \tilde{\mathbf{u}}_{N^e}^{f'}\right] \in \mathbb{R}^{N_1 \times N^e}, \tag{5.4}$$

with N^e being the ensemble size. Note that this matrix will have several vacant entries, namely, those corresponding to the cells with missing mesh points for each ensemble member. Thus, this

matrix will appear in the form of

$$\mathbf{E}^{f'} = \begin{bmatrix} \tilde{u}_{11}^{f'} & \tilde{u}_{12}^{f'} & \star & \dots & \tilde{u}_{1N^e}^{f'} \\ \tilde{u}_{21}^{f'} & \star & \tilde{u}_{23}^{f'} & \dots & \tilde{u}_{2N^e}^{f'} \\ \star & \tilde{u}_{32}^{f'} & \tilde{u}_{33}^{f'} & \dots & \star \\ & \dots & & & \\ \tilde{u}_{N1}^{f'} & \star & \tilde{u}_{N3}^{f'} & \dots & \tilde{u}_{NN^e}^{f'} \end{bmatrix}, \tag{5.5}$$

where $\star$ denotes a missing entry.

We will need to introduce some new notation to compute the ensemble statistics for this method. For each $1 \leq j \leq N_1$, let

$$I_j = \{n \mid \mathbf{x}_n \text{ has a mesh point in cell } L_j\}. \tag{5.6}$$

Then let $N_j^e = |I_j|$. Then we can compute the forecast ensemble mean by

$$\bar{\tilde{\mathbf{u}}}^{f'} = \begin{pmatrix} \frac{1}{N_1^e} \sum_{n \in I_1} \tilde{\mathbf{u}}_{1j}^{f'} \\ \frac{1}{N_2^e} \sum_{n \in I_2} \tilde{\mathbf{u}}_{2j}^{f'} \\ \dots \\ \frac{1}{N_N^e} \sum_{n \in I_N} \tilde{\mathbf{u}}_{Nj}^{f'} \end{pmatrix} \tag{5.7}$$

We will let $\bar{\tilde{u}}_j^{f'}$ denote the j-th component of $\bar{\tilde{\mathbf{u}}}^{f'}$.

The vacant entries in $\mathbf{E}^{f'}$ mean we cannot compute the anomaly matrix directly, as before. The vacant entries must be filled by something in order to subtract the mean matrix. If an entry in the j-th row of $\mathbf{E}^{f'}$ is vacant, then we replace it with the mean of the existing entries from the j-th row. The ensemble matrix then takes the form

$$\tilde{\mathbf{E}}^{f'} = \begin{bmatrix} \tilde{u}_{11}^{f'} & \tilde{u}_{12}^{f'} & \bar{\tilde{u}}_1^{f'} & \dots & \tilde{u}_{1N^e}^{f'} \\ \tilde{u}_{21}^{f'} & \bar{\tilde{u}}_2^{f'} & \tilde{u}_{23}^{f'} & \dots & \tilde{u}_{2N^e}^{f'} \\ \bar{\tilde{u}}_3^{f'} & \tilde{u}_{32}^{f'} & \tilde{u}_{33}^{f'} & \dots & \bar{\tilde{u}}_3^{f'} \\ & \dots & & & \\ \tilde{u}_{N1}^{f'} & \bar{\tilde{u}}_N^{f'} & \tilde{u}_{N3}^{f'} & \dots & \tilde{u}_{NN^e}^{f'} \end{bmatrix} \tag{5.8}$$

This is the key difference between the method described in this chapter and that described in the previous chapter. In the HR case in the previous method, we interpolated to fill in the vacant entries. In this way, each ensemble member contributed to every entry of the ensemble matrix, matrix of means, and anomaly matrix; some just did so indirectly, through interpolation. In this method, we avoid contributions to cell L_j of ensemble members without mesh points present in cell L_j. This introduces an interesting tradeoff: either we "create" information using interpolation, and thus are justified in dividing by the full ensemble size N^e at each step of the process; or we do not fill in using interpolation. We fill in vacant entries using the mean of the present ensemble members, but this does not create any new information because the corresponding entries will vanish in the anomaly matrix. Ensemble members not present in cell L_j do not contribute to the mean, and therefore to the ensemble or anomaly matrix, for cell L_j. We must divide by the correct ensemble size for each row, which complicates the filter calculations and raises questions relating to linearity; we will explore these below.

Let $\tilde{\mathbf{E}}^{f'}$ denote the ensemble matrix, with vacant entries filled in with the means. Then the matrix of anomalies can be computed as before, by subtracting the matrix of means from the matrix of ensemble members. However, we must scale each row by the correct ensemble size. Thus, the scaled anomaly matrix takes the form

$$\mathbf{X}^{f'} = \begin{bmatrix} \frac{1}{\sqrt{N_1^e-1}}\left(\tilde{u}_{11}^{f'} - \bar{\tilde{u}}_1^{f'}\right) & \frac{1}{\sqrt{N_1^e-1}}\left(\tilde{u}_{12}^{f'} - \bar{\tilde{u}}_1^{f'}\right) & \dots & \frac{1}{\sqrt{N_1^e-1}}\left(\tilde{u}_{1N^e}^{f'} - \bar{\tilde{u}}_1^{f'}\right) \\ \frac{1}{\sqrt{N_2^e-1}}\left(\tilde{u}_{21}^{f'} - \bar{\tilde{u}}_2^{f'}\right) & 0 & \dots & \frac{1}{\sqrt{N_2^e-1}}\left(\tilde{u}_{2N^e}^{f'} - \bar{\tilde{u}}_2^{f'}\right) \\ 0 & \frac{1}{\sqrt{N_3^e-1}}\left(\tilde{u}_{32}^{f'} - \bar{\tilde{u}}_3^{f'}\right) & \dots & 0 \\ & \dots & & \\ \frac{1}{\sqrt{N_N^e-1}}\left(\tilde{u}_{N1}^{f'} - \bar{\tilde{u}}_N^{f'}\right) & 0 & \dots & \frac{1}{\sqrt{N_N^e-1}}\left(\tilde{u}_{NN^e}^{f'} - \bar{\tilde{u}}_N^{f'}\right) \end{bmatrix} \tag{5.9}$$

We then proceed with the stochastic ensemble Kalman filter (EnKF). We use the anomaly matrix to compute the forecast covariance and spread. In order to compute the Kalman gain, we must normalize the rows of the ensemble matrix and matrix of means by corresponding ensemble sizes N_j^e *before* applying the observation operator to each of them. This introduces the linearity issue: the observation operator is nonlinear, but we must divide by the effective ensemble sizes before we apply the observation operator. We believe this negatively affects the results of the EnKF

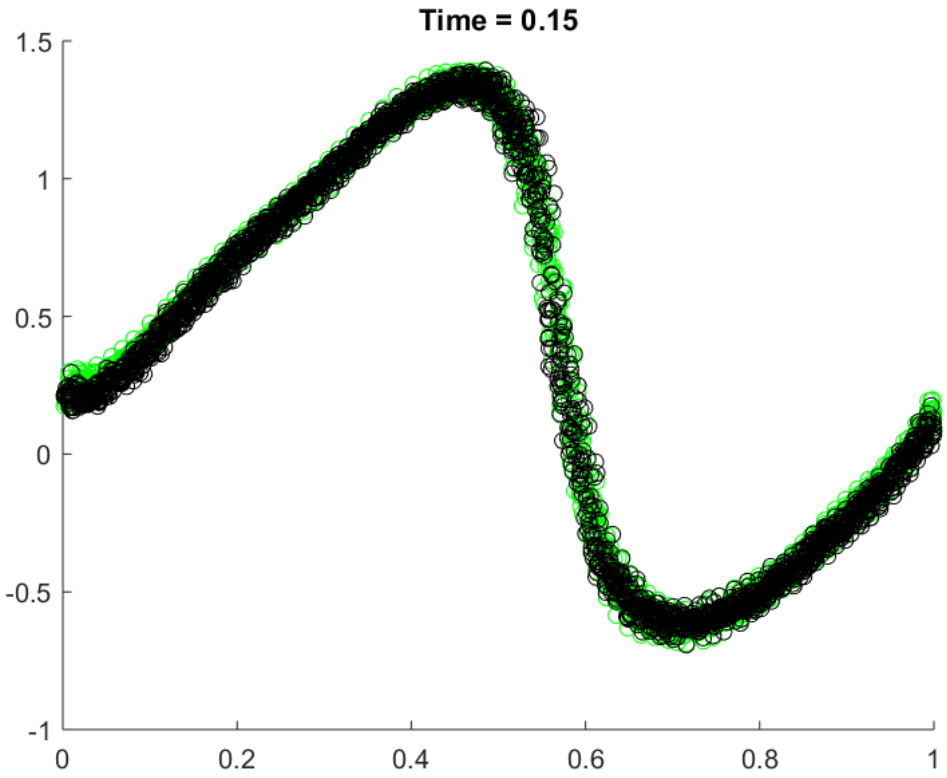


Figure 5.1: Plot of forecast and analysis ensemble members at Time = 0.15, after three data assimilation steps. Forecast ensemble members are in green, analysis are in black.

to a large extent.

5.3 Experimental Setup

We use essentially the same setup for Burgers' equation as we did in the previous chapter. Here, however, we perform data assimilation at every $t = 0.05$ and run the experiment up to final time $T = 0.5$. We use $N_e = 100$ ensemble members; this is significantly higher than in the previous chapter, but it turns that this is necessary in order to have even somewhat reasonable results.

5.4 Results

The most striking result is the presence of a growing error near the shock. For the first few time steps, everything proceeds as planned. The analysis does an excellent job of estimating the truth, with the analysis spread narrower than the forecast spread. This is shown in Figure 5.1.

At the next assimilation step, the method starts to go awry. We can see in Figure 5.2 that along most of the spatial domain, the analysis mean satisfactorily estimates the true state and has a

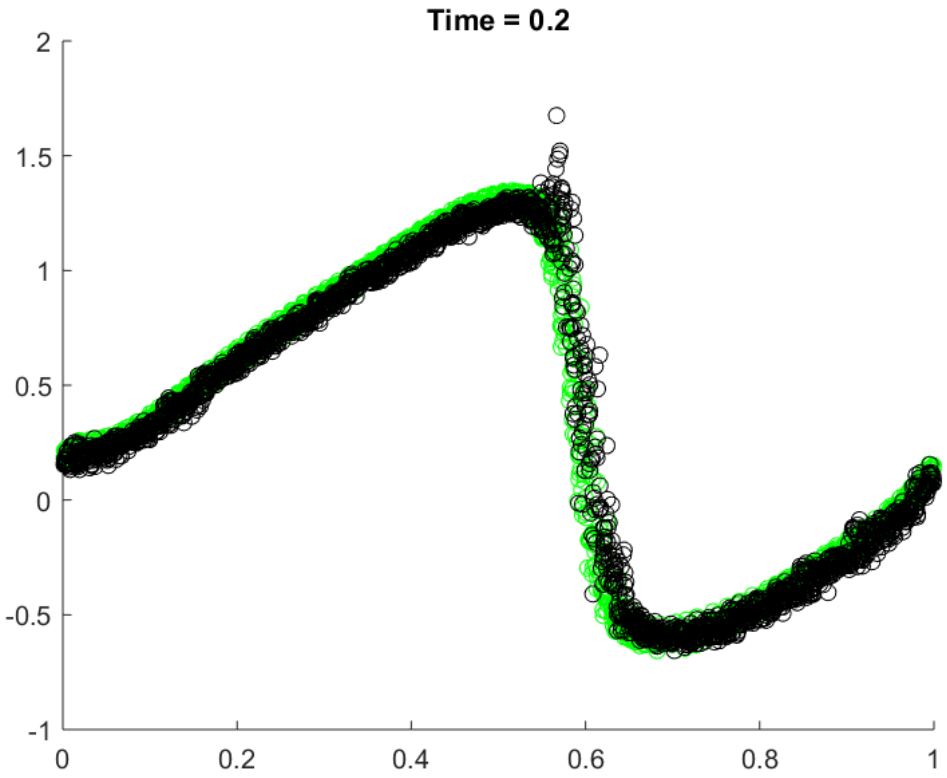


Figure 5.2: Plot of forecast and analysis ensemble members at Time = 0.30, after the fourth data assimilation step. Forecast ensemble members are in green, analysis are in black. We can clearly see here how the analysis overestimates the state along the shock.

narrower spread than the forecast, as expected. This is not the case near the shock. We can see that the analysis significantly overestimates the true state in this region.

This effect seems to continue. Looking at a future time in Figure 5.3, the analysis error is significantly more pronounced. The analysis overestimates the truth towards the left side of the shock, and underestimates near the right side of the shock. In addition, we can see some areas away from the shock where the analysis does a worse job of estimating the truth than before.

These results lend credence to the idea that filling in the gaps in the ensemble members using interpolation, as was done in Chapter 4, is a more promising method than the one employed in this chapter. We suspect that part of the issue is the varying number of ensemble members present in each cell.

The observation, forecast, and analysis plots are shown in Figure 5.4. We can see that both the forecast and analysis RMSE are significantly higher throughout the experiment than the

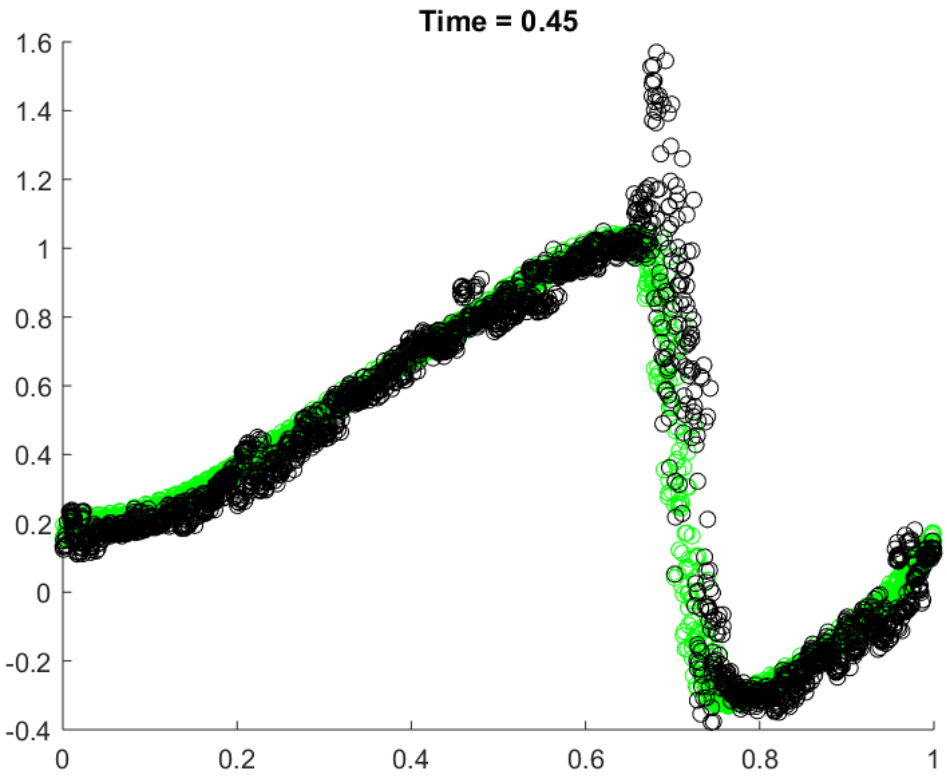


Figure 5.3: Plot of forecast and analysis ensemble members at Time = 0.45, after nine data assimilation steps. Forecast ensemble members are in green, analysis are in black.

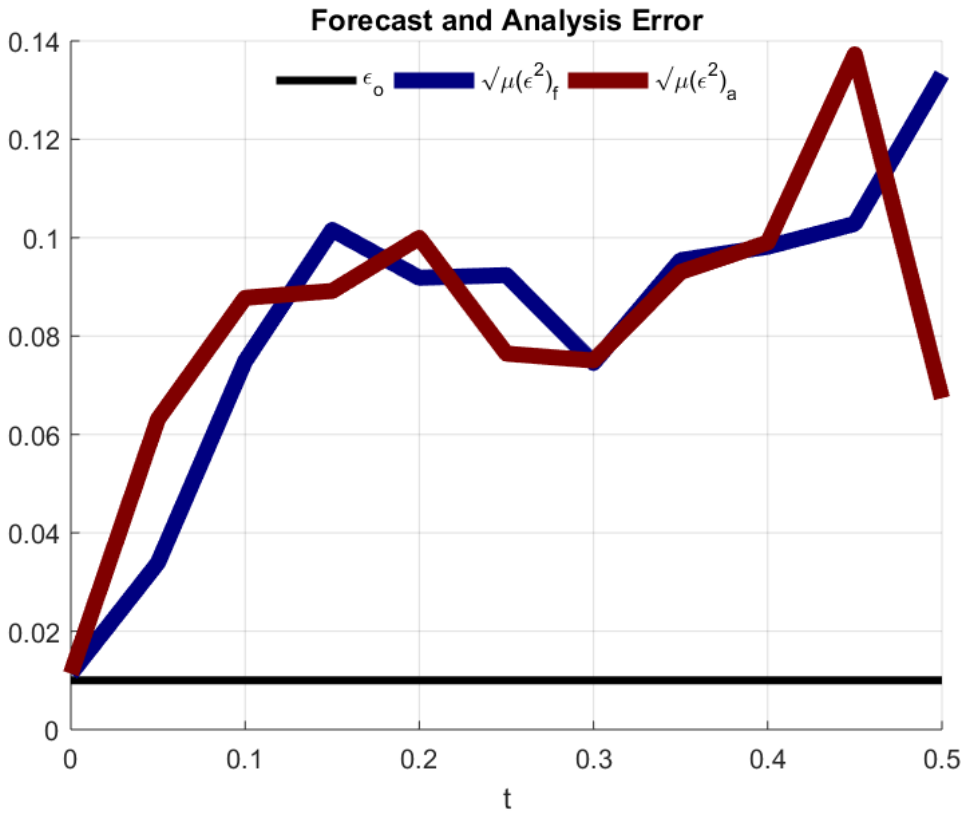


Figure 5.4: Plot of forecast, analysis, and observation root mean square error.

observational error. We also see that it is not in general true that the analysis error is below the forecast error, so assimilating the observations does necessarily help the forecast. These show that the varying ensemble sizes have a large negative impact on the accuracy of this algorithm.

5.5 Conclusions

The method in this chapter is ineffective, unlike the interpolation method from Chapter 4, which worked very well. Interpolating to fill in the gaps in the ensemble members is a much better option than not interpolating and using different effective ensemble sizes. It is likely that this failure is due to the nonlinear nature of the observation operator, and it is possible that a scenario with a linear observation operator, or one where the effective ensemble sizes did not vary so much, would produce better results.

One remedy is increasing the ensemble size will improve the performance of this algorithm. However, in the experiment we look at, we already used a very large ensemble size. It is unlikely that increasing the ensemble size will have much of an impact, and this is supported by preliminary

results. In fact, a drawback to this is that it may introduce even more variation in the effective ensemble sizes being used at the analysis step, which we believe to be a main source of error already. This method needs to be refined much more in order to be effective. Making the remeshing criteria smaller, which means more mesh points will be present in each ensemble member, would make the algorithm more effective.

CHAPTER 6

Adaptive Mesh EnKF on a Moving Reference Mesh

6.1 Introduction

Thus far, the models we have encountered have had a non-conservative adaptive mesh. This was motivated by the characteristics of the neXtSIM model; the number of mesh points is not conserved, introducing an issue that must be addressed. We have discussed two methods for performing ensemble DA on non-conservative adaptive moving mesh models, both of which can be extended to higher-dimensional scenarios.

However, most adaptive moving mesh models involve conservative adaptive moving meshes; that is, the number of mesh points does not change over time, as there is no inserting or removing of mesh points [5]. This is a simpler scenario, and one that has been addressed [21]. Specifically, the dimension of the state space is constant, allowing for a consistent analysis update without the need to project onto a reference mesh, as we have done in the non-conservative case. We deviate from [21] in that we still project onto a reference mesh, which is derived from the previous analysis mean.

For the data assimilation methods developed in this chapter, the movement of the mesh will be driven by an equidistribution condition of a suitable monitor function. A typical example, and the one that we use in the one-dimensional case, is to move the mesh points so that the arc-length of the solution is equal for each pair of consecutive mesh points. This arc-lenth is computed using a spline interpolation of each ensemble member. Thus, the physical and mesh variables are coupled, as was the case for the non-conservative mesh; however, the mesh points here are not Lagrangian. Their velocity is not given simply by the value of the physical variable, but instead by a function of the arc length. We also use Dirichlet boundary conditions, as opposed to periodic boundary conditions.

In this chapter, we describe an implementation of the Local Ensemble Transform Kalman Filter (LETKF) to a conservative adaptive moving mesh model [6]. Use of the LETKF introduces a key localization aspect to our DA algorithm, one that was not present in the non-conservative mesh case. In particular, one parameter that we tune is the "localization radius" of our DA algorithm.

We still keep many of the aspects of our previous experiments, but the use of a conservative mesh and established DA method allow for more aspects of the approach to be tested, i.e. localization radius, rather than just the ones for the interpolation and no-interpolation methods.

6.2 Adaptive Mesh Movement in 1D

The type of mesh movement we describe here is conservative; that is, the number of mesh points for the truth (and for each ensemble member in the DA scheme) does not change over time. Non-conservative adaptive moving meshes are much less common, and are generally used for very specific purposes. Conservative meshes are more versatile, although do not capture the properties of neXtSIM that motivated the work in Chapter 4 and Chapter 5.

We proceed by describing the principles behind adaptive mesh movement as in [5]. The main question that must be answered is this: to most efficiently approximate a function $u(x)$ from its values at a finite number of points, what is the best way to choose the locations of these points? The general answer is to choose a mesh density function, $\rho(x)$, that is somehow related to the numerical error in approximating u, and choose the mesh points so that this density function is equidistributed. The result is that we place the mesh points in such a way that the distance between them is small where the value of $\rho(x)$ is large, and vice versa. We choose the mesh density function $\rho(x)$ in a way so as to minimize interpolation error.

More specifically, for a continuous function $\rho(x) > 0$ on an interval $[a, b]$, and a specified number of mesh points $N > 1$, equidistribution entails finding a mesh $x_1 = a < x_2 < \cdots < x_N = b$ that evenly distributes ρ along the subintervals (x_{i-1}, x_i) determined by the mesh points, so that

$$\int_{x_1}^{x_2} \rho(x)dx = \cdots = \int_{x_{N-1}}^{x_N} \rho(x)dx. \tag{6.1}$$

In other words, we want the area under $\rho(x)$ to be the same over each subinterval. The square of this function ρ is referred to as the *monitor function*, and the function ρ itself is called the *mesh density function*. There are several choices for such a function, but we choose arc length, which is quite commonly used [5].

It can be shown that for a given integer $N > 0$, there exists a unique mesh satisfying (6.1). It can also be shown that an adaptive mesh satisfying the equidistribution condition satisfies certain optimality conditions [5]. Even though we can guarantee theoretical existence of this

equidistributing mesh, we often cannot find it in practice because the integrals in (6.1) cannot be computed analytically. Thus, we must rely on numerical methods to compute the mesh.

Using a coordinate transformation, we can transform the equidistribution condition into a partial differential equation that the moving mesh must satisfy, referred to as the moving mesh PDE (MMPDE) [5]. Standard numerical methods for solving partial differential equations (finite difference, finite element) can be used to solve the MMPDE and advect the moving mesh forward in time. Thus, the equidistribution condition can be satisfied and the optimality of the adaptive mesh, in that arc length is equidistributed, can be approximately preserved throughout the process.

6.3 The Local Ensemble Transform Kalman Filter (LETKF) for a Conservative Adaptive Moving Mesh Model

The adaptive mesh we have just described is *conservative*; i.e., the number of mesh points is preserved throughout. We now describe an algorithm for performing ensemble data assimilation where the ensemble members have physical values defined on conservative adaptive meshes. This is a natural fit for the problem of estimating the true state of a model also defined on a conservative adaptive moving mesh. We implement a version of the LETKF to introduce localization to the process. The fact that all of the ensemble members will have the same number of mesh points at all times avoids many of the complications that arise when dealing with non-conservative meshes. Specifically, the dimension of the state space in the data assimilation problem is constant in the case of a conservative mesh.

Data assimilation is performed by projecting all of the ensemble members onto a common mesh. This mesh is generated as the optimal, equidistributing mesh for the ensemble mean of the previous step. The analysis update is then performed on this common mesh, after which the ensemble members are projected back onto their original meshes. The common mesh for the next analysis step is then computed from the new analysis ensemble mean. Because the common mesh is computed using the equidistribution condition, it is of course adaptive and will evolve over time.

We suppose that the adaptive mesh is defined in the domain $[0, L]$, and that the endpoints are always the first and last points of the mesh; they are fixed in time, and are the only such mesh points.

6.3.1 The model state and its evolution

The model is solved on a conservative adaptive mesh, whose evolution is coupled with that of the physical value(s) defined at those mesh points. Thus, it is natural to augment the adaptive mesh points to the physical values in the state vector. If there are N mesh points, then the dimension of the state vector will be $2N$:

$$\mathbf{x} = (u_1, u_2, \ldots, u_N, z_1, z_2, \ldots, z_N) \in \mathbb{R}^N \times [0, L]^N, \tag{6.2}$$

where the z_i are the adaptive mesh points and the u_i are the physical values defined at the z_i. It is key to remember here that N is constant, as there is no remeshing process at work.

The model operates between observation times, and will depend on time if the PDE underlying the model is non-autonomous. We consider both the PDE that the physical values must satisfy and the moving mesh PDE as part of the model evolution. Thus, the model applies to the entire state vector $\mathbf{x}$, both the u_i and the z_i. If $\mathbf{x}_k = \mathbf{x}(t_k)$, then we can write the model evolution from time t_k to time t_{k+1} as the mapping

$$\mathbf{x}_{k+1} = \mathcal{M}(\mathbf{x}_k). \tag{6.3}$$

Again, this model $\mathcal{M}$ will implement the MMPDE, but will not implement any remeshing process as in the other chapters.

6.3.2 Mapping onto a fixed reference mesh

Given a state vector $\mathbf{x} = (u_1, u_2, \ldots, u_N, z_1, z_2, \ldots, z_N) \in \mathbb{X}_N = \mathbb{R}^N \times [0, L]^N$, we define a map $\mathcal{P} : \mathbb{X}^N \to \mathbb{X}^N$ that projects each ensemble member (and the truth) onto a common mesh. At each time step, this common mesh is the equidistributing mesh of the analysis mean of the previous time step. Specifically, the adaptive mesh is mapped to the common reference mesh, and the physical values are interpolated from the adaptive mesh points to the common mesh points.

A basic case, in which only two ensemble members are used for illustration, is shown in Figure 6.1. The reference mesh on which the analysis mean is computed is taken as the common equidistributing mesh of the ensemble mean, which is computed from the analysis ensemble members of the previous time step. This does introduce a "lag" in the common mesh, as it is working from the previous time step instead of the current time step. This lag does seem to have an impact on the data assimilation

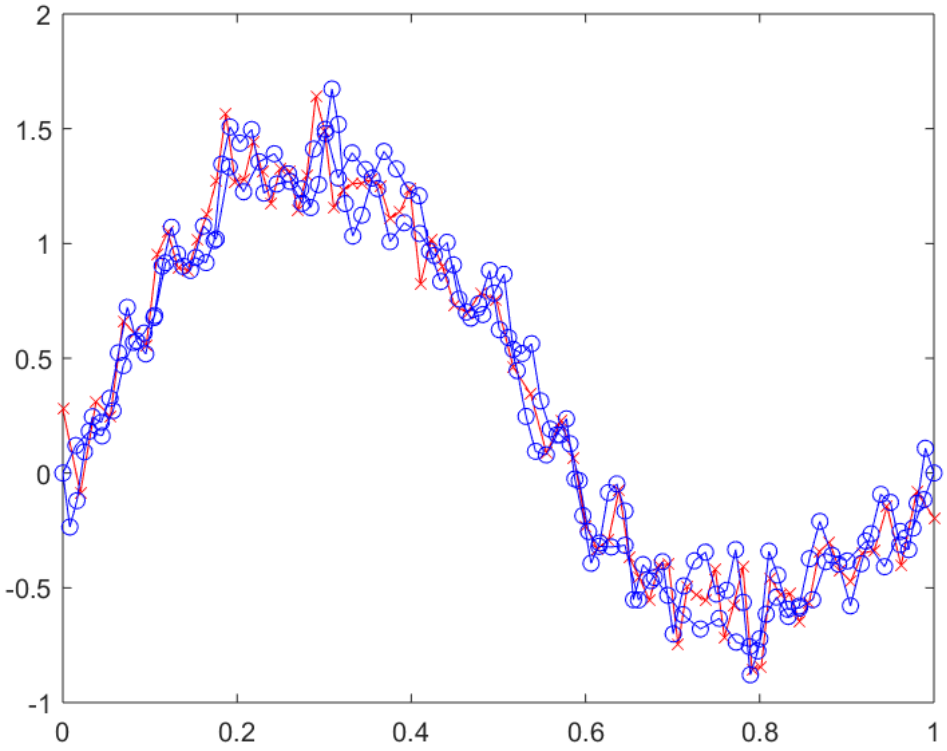


Figure 6.1: Illustration of projection onto the common mesh for the data assimilation update. Two ensemble members are shown in blue; a common mesh is computed from their ensemble mean, which is shown in red.

effectiveness, at first, but it appears to be resolved fairly quickly, as we see below. A method that computes the reference mesh a different way, namely as the intersection of metric tensors, is the subject of other work [45].

We do not update the mesh points in the analysis step, so we work on a reduced state vector $\hat{\mathbf{u}}$ of dimension N. Because the first and last mesh points are fixed throughout time, and because the mesh from the previous time step is integral to the process, it does not make much sense to consider updating the mesh points here. The case for updating mesh points is more intriguing when the boundary is not fixed [21], as then there is more to estimate (i.e., the boundary location).

6.4 Experimental Setup

To test the efficacy of this data assimilation algorithm, we again consider Burgers' equation, with the same initial condition as in Chapter 3. However, this time we take the boundary conditions to be Dirichlet. We let our observations be Eulerian, i.e., the observation locations are fixed in

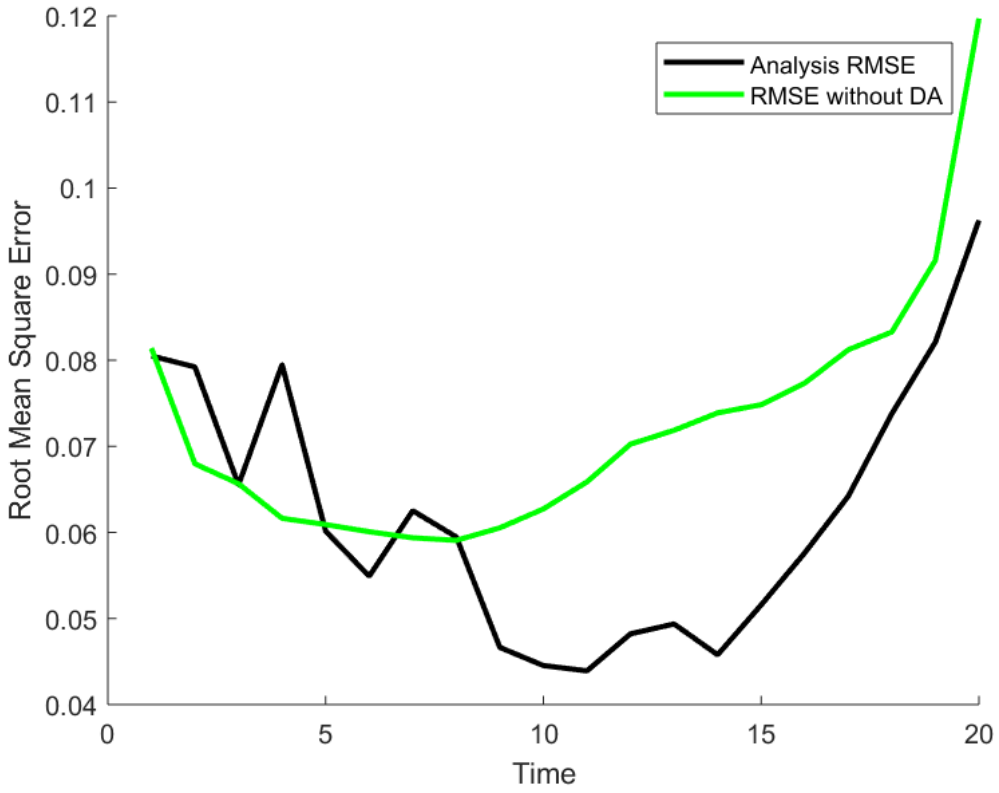


Figure 6.2: Plot of the analysis root mean square error and the RMSE of a free run for the conservative adaptive mesh case. The horizontal axis shows the time step.

time. We use 61 adaptive mesh points for each ensemble member, and make observations at 10 fixed locations. We take the observation error to be 5% and the initial perturbation to be 20%. We take an ensemble size of just 5 members and run the experiment up to time $T = 2$, with time 0.1 between observations. The inflation factor used is $\rho = 1.5$. The localization radius is taken to be 3.

6.5 Results

The results of this experiment are shown in Figure 6.2, in which the analysis RMSE and the RMSE for a "free" run are shown, that is, one in which no observations are assimilated. We see that assimilating the Eulerian observations does perform better overall than the DA-free run. It does take some time for the analysis RMSE to pass below the DA-free RMSE, but it seems to perform better after about the eighth time step. This is because of the lag in the common mesh. There does seem to be an uptick in the RMSE late in the process. It may be that the increasing sharpness of the moving front may be hard for the algorithm to follow.

6.6 Conclusions

This adaptation of the LETKF to a conservative adaptive moving mesh model works quite well. It makes sense to use an LETKF for this problem, as the physical values at adjacent locations are likely to be highly correlated. There are several parameters that need to be tuned, including localization radius and inflation factor. It also needs to be tested on models of higher complexity. An application of this method to the two-dimensional Fitzhugh-Nagumo equation is in progress.

One possible direction to take is to compute the reference mesh differently. In ongoing work in [46], the common mesh is computed using a metric tensor, instead of as an equidistributing mesh for the previous analysis. A significant advantage this could offer is eliminating the lag introduced by using a common mesh computed using a previous time step.

Another interesting direction to take is to try to adapt the LETKF for a non-conservative adaptive moving mesh model. This introduces much more difficulty; the methods used in this chapter have only been developed for meshes with a fixed number of mesh points. It would be a significant challenge to combine these two aspects, but it may be worthwhile to do so in order to take advantage of the LETKF.

CHAPTER 7

Summary of Results and Future Directions

The common thread among the data assimilation methods developed here is their suitability for implementation on adaptive moving meshes; indeed this is built into their structure. They do, however, vary significantly in their methodology, implementation, and efficacy.

Based on the results of the interpolation method from Chapter 4 and the no-interpolation method from Chapter 5, it is clear that the interpolation method should be the method of choice moving forward. It introduces far less error than the method with no interpolation and handles the nonlinearity much more effectively. This is what should be used going forward in higher-dimensional models, particularly neXtSIM.

The differences in the implementations of an adaptive moving mesh EnKF for the interpolation method and no-interpolation method have large consequences. It is possible that the no-interpolation method could be suitable for a model that is both much more linear and does not have much variation in the effective ensemble sizes being used. The difference in effective ensemble sizes leads to significant variation in the scaling factors for the anomaly matrix. It is the evolution of the sharp shock in Burgers' equation that leads to the differing effective ensemble sizes. Thus, it is natural that an equation where the physical values do not exhibit such a shock may be called for.

The method from Chapter 4 does work very effectively for both Burgers' equation and the Kuramoto-Sivashinsky equation. Testing on a model like KSM was necessary, as the form of Burgers' equation we used was dissipative, meaning the data assimilation problem would not remain interesting for very long. The parameter values we used for KSM led the model to exhibit chaotic behavior, which the keeps the data assimilation problem from being trivial indefinitely.

Extending the method from Chapter 4 to a two-dimensional model, and then to neXtSIM, is a natural progression of this work. Some issues that could arise are sparsity of mesh points and increased computational complexity. The framework of the algorithm extends naturally to higher dimensions; the issues we encounter in this leap are more practical.

The method in Chapter 6 exhibits promising results for Burgers' equation. The implementation discussed here is for conservative mesh models. This was achieved because it incorporates the equidistribution concept, which leads to a conservative mesh model in the literature. It would certainly be interesting to apply this to higher-order models. Another challenging direction would be combine this method with that of Chapter 4, which would combine the LETKF with a nonconservative adaptive moving mesh model.

REFERENCES

[1] J. Burgers, "A mathematical model illustrating the theory of turbulence," vol. 1 of *Advances in Applied Mechanics*, pp. 171 – 199, Elsevier, 1948.

[2] H. Stommel, "Thermohaline convection with two stable regimes of flow," *Tellus A: Dynamic Meteorology and Oceanography*, vol. 13, no. 2, pp. 224–230, 1961.

[3] V. Dansereau, J. Weiss, P. Saramito, and P. Lattes, "A maxwell elasto-brittle rheology for sea ice modelling," *The Cryosphere*, vol. 10, no. 3, pp. 1339–1359, 2016.

[4] K. Emanuel, "Tropical cyclones," *Annual Review of Earth and Planetary Sciences*, vol. 31, pp. 75–104, 2003.

[5] W. Huang and R. D. Russell, *Adaptive moving mesh methods*, vol. 174. Springer Science & Business Media, 2010. ISBN:978-1-4419-7916-2.

[6] B. Hunt, E. Kostelich, and I. Szunyogh, "Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter," *Physica D*, vol. 230, pp. 112–126, 2007.

[7] L. Slivinski, E. Spiller, A. Apte, and B. Sandstede, "A hybrid particle-ensemble kalman filter for lagrangian data assimilation," *Monthly Weather Review*, vol. 143, pp. 195–210, 2015.

[8] H. Weller, T. Ringler, M. Piggott, and N. Wood, "Challenges facing adaptive mesh modeling of the atmosphere and ocean," *Bulletin of the American Meteorological Society*, vol. 91, pp. 105–108, 2010.

[9] M. J. Baines, M. E. Hubbard, and P. K. Jimack, "Velocity-based moving mesh methods for nonlinear partial differential equations," *Communications in Computational Physics*, vol. 10, no. 3, pp. 509–576, 2011.

[10] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484 – 512, 1984.

[11] W. Huang, L. Zheng, and X. Zhan, "Adaptive moving mesh methods for simulating one-dimensional groundwater problems with sharp moving fronts," *International Journal for Numerical Methods in Engineering*, vol. 54, no. 11, pp. 1579–1603, 2002.

[12] A. Alharbi and S. Naire, "An adaptive moving mesh method for thin film flow equations with surface tension," *Journal of Computational and Applied Mathematics*, vol. 319, pp. 365 – 384, 2017.

[13] C. Pain, M. Piggott, A. Goddard, F. Fang, G. Gorman, D. Marshall, M. Eaton, P. Power, and C. de Oliveira, "Three-dimensional unstructured mesh ocean modelling," *Ocean Modelling*, vol. 10, no. 1, pp. 5 – 33, 2005. The Second International Workshop on Unstructured Mesh Numerical Modelling of Coastal, Shelf and Ocean Flows.

[14] D. R. Davies, C. R. Wilson, and S. C. Kramer, "Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics," *Geochemistry, Geophysics, Geosystems*, vol. 12, no. 6, 2011.

[15] A. Budhiraja, E. Friedlander, C. Guider, C. Jones, and J. Maclean, "Assimilating data into

models," 2018.

[16] M. Asch, M. Bocquet, and M. Nodet, *Data Assimilation: Methods, Algorithms, and Applications*. Fundamentals of Algorithms, SIAM, Philadelphia, 2016. ISBN:978-1-611974-53-9.

[17] A. Carrassi, M. Bocquet, L. Bertino, and G. Evensen, "Data assimilation in the geosciences: An overview of methods, issues, and perspectives," *Wiley Interdisciplinary Reviews: Climate Change*, vol. 9, no. 5, p. e535, 2018.

[18] F. Fang, M. Piggott, C. Pain, G. Gorman, and A. Goddard, "An adaptive mesh adjoint data assimilation method," *Ocean Modelling*, vol. 15, no. 1, pp. 39 – 55, 2006. The Third International Workshop on Unstructured Mesh Numerical Modelling of Coastal, Shelf and Ocean Flows.

[19] G. Evensen, *Data Assimilation: The Ensemble Kalman Filter*. Springer-Verlag/Berlin/Heildelberg, second ed., 2009. ISBN:978-3-642-03711-5.

[20] P. L. Houtekamer and F. Zhang, "Review of the ensemble kalman filter for atmospheric data assimilation," *Monthly Weather Review*, vol. 144, no. 12, pp. 4489–4532, 2016.

[21] B. Bonan, N. K. Nichols, M. J. Baines, and D. Partridge, "Data assimilation for moving mesh methods with an application to ice sheet modelling," *Nonlinear Processes in Geophysics*, vol. 24, no. 3, pp. 515–534, 2017.

[22] J. Du, J. Zhu, F. Fang, C. Pain, and I. Navon, "Ensemble data assimilation applied to an adaptive mesh ocean model," *International Journal for Numerical Methods in Fluids*, vol. 82, no. 12, pp. 997–1009, 2016.

[23] S. Bouillon, P. Rampal, and E. Olason, "Sea ice modelling and forecasting," in *New Frontiers in Operational Oceanography* (E. P. Chassignet, A. Pascual, J. Tintoré, and J. V. (Eds.)n, eds.), ch. 15, pp. 423–444, GODAE OceanView, 2018.

[24] P. Rampal, S. Bouillon, E. Ólason, and M. Morlighem, "nextsim: a new lagrangian sea ice model," *Cryosphere*, vol. 10, no. 3, 2016.

[25] D. Marsan, H. L. Stern, R. Lindsay, and J. Weiss, "Scale dependence and localization of the deformation of Arctic sea ice," *Phys. Rev. Lett.*, vol. 93, p. 178501, Oct. 2004.

[26] P. Rampal, J. Weiss, D. Marsan, R. Lindsay, and H. Stern, "Scaling properties of sea ice deformation from buoy dispersion analysis," *Journal of Geophysical Research: Oceans*, vol. 113, no. C3, 2008. C03002.

[27] M. Rabatel, P. Rampal, A. Carrassi, L. Bertino, and C. K. Jones, "Impact of rheology on probabilistic forecasts of sea ice trajectories: application for search and rescue operations in the arctic.," *Cryosphere*, vol. 12, no. 3, 2018.

[28] S. Bouillon and P. Rampal, "Presentation of the dynamical core of nextsim, a new sea ice model," *Ocean Modelling*, vol. 91, pp. 23 – 37, 2015.

[29] G. Compère, J.-F. Remacle, J. Jansson, and J. Hoffman, "A mesh adaptation framework for dealing with large deforming meshes," *Int. J. Numer. Meth. Engng.*, vol. 82, pp. 843–867, 2009.

[30] G. Compère, J. F. Remacle, and E. Marchandise, "Transient mesh adaptivity with large rigid-body displacements," in *Proceedings of the 17th International Meshing Roundtable* (R. Garimella, ed.), (Berlin), pp. 213–230, Springer, 2008.

[31] D. Partridge, *Numerical modelling of glaciers: moving meshes and data assimilation.* PhD thesis, University of Reading, 2013.

[32] J. Maddison, D. Marshall, C. Pain, and M. Piggott, "Accurate representation of geostrophic and hydrostatic balance in unstructured mesh finite element ocean modelling," *Ocean Modelling*, vol. 39, no. 3, pp. 248 – 261, 2011.

[33] P. Farrell, M. Piggott, C. Pain, G. Gorman, and C. Wilson, "Conservative interpolation between unstructured meshes via supermesh construction," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 33, pp. 2632–2642, 2009.

[34] P. K. Jain, K. Mandli, I. Hoteit, O. Knio, and C. Dawson, "Dynamically adaptive data-driven simulation of extreme hydrological flows," *Ocean Modelling*, vol. 122, pp. 85–103, 2018.

[35] I. Babuška and A. Aziz, "On the angle condition in the finite element method," *SIAM Journal on Numerical Analysis*, vol. 13, no. 2, pp. 214–226, 1976.

[36] P. H. Saksono, W. G. Dettmer, and D. Perić, "An adaptive remeshing strategy for flows with moving boundaries and fluid-structure interaction," *International Journal for Numerical Methods in Engineering*, vol. 71, no. 9, pp. 1009–1050, 2007.

[37] M. Verlaan and A. W. Heemink, "Nonlinearity in data assimilation applications: A practical method for analysis," *Monthly Weather Review*, vol. 129, no. 6, pp. 1578–1589, 2001.

[38] O. Pannekoucke, M. Bocquet, and R. Ménard, "Parametric covariance dynamics for the nonlinear diffusive burgers equation," *Nonlinear Processes in Geophysics*, vol. 25, no. 3, pp. 481–495, 2018.

[39] D. T. Papageorgiou and Y. S. Smyrlis, "The route to chaos for the kuramoto-sivashinsky equation," *Theoretical and Computational Fluid Dynamics*, vol. 3, no. 1, pp. 15–42, 1991.

[40] G. Burgers, P. Jan van Leeuwen, and G. Evensen, "Analysis scheme in the ensemble kalman filter," *Monthly weather review*, vol. 126, no. 6, pp. 1719–1724, 1998.

[41] P. Sakov and P. R. Oke, "A deterministic formulation of the ensemble Kalman filter: an alternative to ensemble square root filters," *Tellus, Ser. A*, vol. 60, no. 2, pp. 361–371, 2008.

[42] "Data assimilation using adaptive, non-conservative, moving mesh models,"

[43] J. L. Anderson and S. L. Anderson, "A monte carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts," *Monthly Weather Review*, vol. 127, no. 12, pp. 2741–2758, 1999.

[44] P. N. Raanes, M. Bocquet, and A. Carrassi, "Adaptive covariance inflation in the ensemble kalman filter by gaussian scale mixtures," *arXiv preprint arXiv:1801.08474*, 2018.

[45] C. Guider, C. Krause, N. Shankar, and E. Van Vleck, "Data assimilation with adaptive moving

meshes: An interpolation approach," 2019.

[46] W. Huang, C. Krause, D. Mechem, E. Van Vleck, and M. Zhang, "A metric tensor approach to data assimilation on adaptive moving meshes," 2019.