UNIVERSITY OF CALIFORNIA SAN DIEGO

Data Assimilation and Precision Annealing Monte Carlo Method in Nonlinear Dynamical Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Physics

by

Kangbo Hao

2020

The dissertation of Kangbo Hao is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I want to thank my advisor, Henry Abarbanel, for his tremendously helpful insight, advice and support. Henry has always been guiding me through this long journey of graduate study and providing valuable feedbacks. Without his guidance, I can never accomplish what I did.

I want to thank my committee member: Professors Philip E. Gill, Ken Intriligator, Gabriel A. Silva, and Congjun Wu, for their time and effort.

I would like to thank Chris White and his team at Microsoft for their support.

I would also like to thank Zhe An, Zheng Fang and Adrian Wong for the inspiring discussions and collaborations. Many thanks to all other colleagues in the group for the stimulating discussions, questions and comments. You all together made my 4 years in the group a really enjoyable experience.

Finally, I want to thank all my families and friends, for their endless support. I cannot be who I am now without any of you.

Chapter 2-4, in part, has been adapted from Z. Fang, A. S. Wong, K. Hao, A. J. Ty, and H. D. Abarbanel. Precision annealing monte carlo methods for statistical data assimilation and machine learning. *Physical Review Research*, 2(1):013050, 2020 and A. S. Wong, K. Hao, Z. Fang, and H. D. Abarbanel. Precision Annealing Monte Carlo Methods for Statistical Data Assimilation: Metropolis-Hastings Procedures. arXiv preprint arXiv:1901.04598 (2019). The author of this thesis was a co-first author of these papers.

VITA

2014	B. S. in Physics *magna cum laude*, University of Illinois at Urbana-Champaign, Urbana
2014-2019	Graduate Teaching Assistant, University of California San Diego
2020	Ph. D. in Physics, University of California San Diego

PUBLICATIONS

Z. Fang, A. S. Wong, K. Hao, A. J. Ty, and H. D. Abarbanel. Precision annealing monte carlo methods for statistical data assimilation and machine learning. *Physical Review Research*, 2(1):013050, 2020.

A. S. Wong, K. Hao, Z. Fang, and H. D. Abarbanel. Precision Annealing Monte Carlo Methods for Statistical Data Assimilation: Metropolis-Hastings Procedures. arXiv preprint arXiv:1901.04598 (2019).

ABSTRACT OF THE DISSERTATION

Data Assimilation and Precision Annealing Monte Carlo Method in Nonlinear Dynamical Systems

by

Kangbo Hao

Doctor of Philosophy in Physics

University of California San Diego, 2020

Professor Henry D. I. Abarbanel, Chair

In the study of data assimilation, people focus on estimating state variables and parameters of dynamical models, and make predictions forward in time, using given observations. It is a method that has been applied to many different fields, such as numerical weather prediction and neurobiology.

To make successful estimations and predictions using data assimilation methods, there are a few difficulties that are often encountered. First is the quantity and quality of the data. In some of the typical problems in data assimilation, the number of observations are usually a few order of magnitude smaller than the number of total variables. Considering this and the fact that almost

all the data gathered are noisy, how to estimate the observed and unobserved state variables and make good predictions using the noisy and incomplete data is one of the key challenge in data assimilation. Another issue arises from the dynamical model. Most of the interesting models are non-linear, and usually chaotic, which means that a small error in the estimation will grow exponentially over time. This property of the chaotic system addresses the necessity of accurate estimations of variables.

In this thesis, I will start with an overview of data assimilation, by formulating the problem that data assimilation tries to solve, and introducing several widely used methods. Then I will explain the Precision Annealing Monte Carlo method that has been developed in the group, as well as its variation using Hamiltonian Monte Carlo. Finally I will demonstrate a few example problems that can be solved using data assimilation methods, varying from a simple but instructional 20-dimension Lorenz 96 model, to a complicated ocean model named Regional Ocean Modeling System.

Chapter 1

Introduction

Systems with complex dynamical behaviors have gained interest in physics over the last century. A well-known example is numerical weather predictions (NWP), the results of which can be found probably every day on TVs or in the weather app on our phones. In order to describe the system and solve some interesting problems, we will formulate a model with a set of mathematical equations, usually a set of differential equations, to represent the evolution of the system and the interactions between variables in the model. However, in many of these systems, our knowledge is still limited and therefore a model representing the real system perfectly is not available. In other cases, we may already know most details of a system from the studies over centuries, but due to constraints like numerical instability or computing time, not all the knowledge can be included in the model. As a result, in most of the study, if not all, we need to exclude the some details of the system that are believed to be less important to the problems we focus on.

Now we have a model of the real system, although a imperfect one, but we are not done yet. Most likely, people are interested in estimating the state of the system and making predictions. Surely we can start at any point within the reasonable dynamical range, and integrating the model forward will give us a path. But we have infinite many possible paths and we do not know which one is the accurate predictions. To make good predictions, besides a relatively good model, we also need to measure the state variables of the model at different time as our data. Then, using these measurements, we can predict forward in time.

Unfortunately, additional problems arises in regard to the data. First, as one can imagine, whatever equipment we used to measure the data, it is almost guaranteed that the data is inaccurate with a certain level of noise. In some cases, as long as the noise is within the tolerance, we can still use the noisy measurements along with the model to make decent predictions. However, when facing a model with complex dynamical behaviors, usually the model is chaotic, and the presence of noise will significantly hinder our ability of predicting, as the error will grow exponentially fast. The properties of chaotic systems will be discussed in more details later in Chapter 2.

Another issue is that the measurements may be incomplete. For example, in a multi-layer ocean model, after discretizing the 3-dimensional space, we usually need the two component of horizontal velocities u, v, the pressure P, the temperature T and the salinity S at each grid point. However, the sources of measurements greatly limit the accessibility of the data. One of the easily accessible data is the satellite data, but it can only provide data at the surface of the ocean. If this is the only source available, then before making predictions, we need to first have a fairly accurate guess to the state variables from the layers in deep ocean, which is often very challenging.

We have gathered two key ingredients of the problem: a imperfect model and a set of noisy, incomplete data within a time window. If our goal is to make predictions after the measurement window, we need to first have good estimates of the state parameters at least at the end of the measurement window, as well as all the unknown or inaccurate parameters in the model.

1.1 A Brief Overview of Data Assimilation

Data assimilation is a field originated from the study of NWP, which could solve the problem we have stated above. Given the estimate of all the necessary state variables at a given time, the study of NWP would use these estimates along with the model to predict the model's evolution in the future. Estimating the current state itself, however, is indeed a complicated problem. In 1963, Lorenz [17] found that the model describing atmosphere is very sensitive to initial conditions, which means the forecast error will grow in time and predictions will fail eventually. This finding gives more importance to the study of state and parameter estimation, as more precise estimations in initial conditions will result in reliable predictions for a longer period of time. In order to find a better initial condition for forecast, people started to incorporate observed data within a time window, and developed a variety of techniques, which are now known as data assimilation, to estimate all the state variables and parameters as accurate as possible. Then forecast can be made after the end of observation window.

The Cressman algorithm [3], a simple interpolation algorithm, is one of the earliest attempt in data assimilation. Since then, benefiting from the progress in both theory and computing capability, many different data assimilation methods have been developed. These methods all fall into two different main categories: sequential assimilation and variational assimilation [29].

In sequential assimilation, starting from the beginning of the observation window, the state variables will be updated as moving towards the end of the window, using the measurements and estimations at current time and from the past. In other words, no measurements will be used to estimate the variables from the time before. The Cressman algorithm mentioned above is an example of sequential assimilation. Other examples of commonly used ones are nudging [10], Kalman filter [11], and ensemble Kalman filter [5].

Variational assimilation, on the other hand, takes a different approach. It will update variables on all the measurements within the observation window, usually by minimizing a cost function or action. In this case, the information from measurements can be carried both forward and backward in time, but the computation cost would typically be higher than sequential assimilation methods. An example of variational assimilation is four dimensional variational (4D-VAR) [21]. In Chapter 3, I will discuss about the precision annealing method, which is another example of variational assimilation methods.

1.2 Overview of the Dissertation

This dissertation contains 6 chapters. This first chapter is a brief introduction to the problems we are trying to solve, with the attempts to address why data assimilation is important. A brief overview of data assimilation is also provided.

The second chapter will explain in detail, the setup of the problems that data assimilation tries to solve. An overview of nonlinear system and chaos is also discussed in this chapter, since it is critical in understanding why we even need data assimilation. Then we will explain several well formulated methods in data assimilation, as well as a probabilistic point of view of data assimilation and its relation to path integrals, which provides some insight to the formulations of the method we will describe in Chapter 3.

Chapter 3 will present the annealing method that has been developed over the past few years within the group, starting with a brief introduction of already established variational annealing method. Then we will discuss in detail of the Precision Annealing Monte Carlo (PAMC) method, and its improvement by replacing Metropolis-Hastings algorithm with Hamiltonian Monte Carlo.

Chapter 4 will introduce an instructional model in data assimilation, the Lorenz 96 model, explain the setup and show the results using PAMC on the model.

Chapter 5 will discuss two of the more complicated model, the shallow water model and Regional Ocean Modeling System (ROMS), as well as some results on those two models, using PAMC method with Hamiltonian Monte Carlo and nudging method respectively.

Chapter 6 is the conclusion chapter, explaining the difficulties we have encountered when moving to the models with higher dimensions and more complicated features, and discuss about some possible improvements in the annealing methods we can explore in future work.

Chapter 2

Problem Formulation

To start describing the problem we are trying to solve in data assimilation, we first need a model for the dynamical system, which usually consists of some differential equations.

$$\frac{dx_a(t)}{dt} = F_a(\mathbf{x}(t), \mathbf{p}) \tag{2.1}$$

$\mathbf{x}(t)$ is a vector of state variables at time t, and $\mathbf{p}$ is a set of parameters. The subscript $a = 1, 2, ..., D$, where D is the number of dimensions of the model. In practice, we would usually discretize the model into a finite number of time steps, $\{t_0, t_1, \ldots, t_F\}$. Then the discretized version of equations 2.1 is

$$x_a(n+1) = f_a(\mathbf{x}(n), \mathbf{p}) \tag{2.2}$$

where $n = 0, 1, \ldots, F$ representing the time step t_n.

Along with the model, we also have a set of measurements $\mathbf{Y} \equiv \{\mathbf{y}(\tau_1), \mathbf{y}(\tau_2), \ldots, \mathbf{y}(\tau_M)\}$ within the time window, at time $t_0 \leq \tau_k \leq t_F$ for $k = 0, 1, \ldots, M$. This setup is illustrated in Figure 2.1 For the rest of the chapter, if the model is discretized in time, we shall denote $\mathbf{y}(\tau_k)$ as $\mathbf{y}(k)$, and $\mathbf{x}(t_n)$ as $\mathbf{x}(n)$ for simplicity. The goal of data assimilation is to transfer information from measurements $\mathbf{Y}$ to the model, or in other words, to synchronize the model to the measurements, so that we could estimate the state variables $\mathbf{x}(t)$ and parameters $\mathbf{p}$.

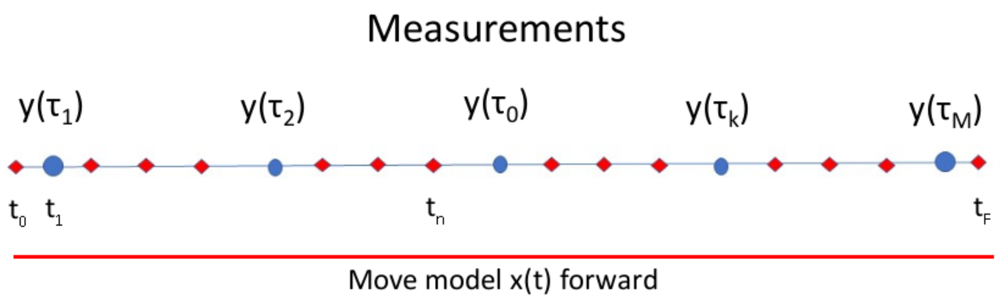


Figure 2.1: A illustration of the problem setup during the time window $t_0 \leq t \leq t_F$. Measurements are made in L dimensions at times $t = \tau_k$, where $0 \leq k \leq M$ and $t_0 \leq \tau_k \leq t_F$. The model in D dimension is integrated forward following equations 2.2.

As we have already introduced in Chapter 1, there are a few issues about the model and measurements I would like to address again here:

1. Among all the parameters in $\mathbf{p}$, some of them may be unknown. Also the model can be inaccurate and has errors itself. One of the typical sources of error would be the discretization in time or space.

2. The measurements are noisy, from the equipment we use to take measurements. Usually, as a reasonable approximation to simplify the problem, we would assume the noise follows Gaussian distribution.

3. The measurements are incomplete. For each $\mathbf{y}(\tau_k)$, there are only L variables measured, where $L < D$ and often $L \ll D$. For example, in a typical meteorology model dealing with real data, the number of measurements is typically two order of magnitude smaller than the total number of variables [12].

Taking these constraint into consideration, we will make a slight modification to Eq. 2.2:

$$x_a(n+1) = f_a(\mathbf{x}(n), \mathbf{p}) + \eta \tag{2.3}$$

where η represents the error in the model. Similarly, we can write the measurements $\mathbf{y}(k)$ as

$$\mathbf{y}(k) = \mathbf{H}\mathbf{x}(k) + \varepsilon \tag{2.4}$$

$\mathbf{H}$ is the observation matrix and ε is the noise in measurements. In many cases, the observation matrix $\mathbf{H}$ is simply a projection matrix, and is very likely not a full rank matrix since the measurements are incomplete.

2.1 Nonlinear Systems and Chaos

In most of the interesting problems, the underlying models are nonlinear, which means that the Eq. 2.1 is a set consisting of non-linear differential equations. To better discuss the behaviors and properties of nonlinear dynamical systems, I will first start explaining some basic concepts.

2.1.1 Trajectory and Phase Space

In general, the state of a nonlinear model at time t can be determined by its state variables $\mathbf{x}(t)$ and their time derivatives $\dot{\mathbf{x}}(t)$. The $\{\mathbf{x}, \dot{\mathbf{x}}\}$ space is called the phase space.

Starting from a position in the phase space as the initial condition, integrating the dynamical equations Eq. 2.1 over a time period, we will find a solution $\mathbf{x}(t)$ as well as $\dot{\mathbf{x}}(t)$, which forms a line in the phase space. Since usually the model has a large number of state variables, the phase space is a high dimensional space. In practice, however, it is useful to project the trajectory down to a two dimensional space $\{x_i, \dot{x}_i\}$. The plot of the trajectory in such space can tell us its behavior even though it may not represent all of its structure in the whole phase space.

If the trajectory passes a point in the phase space where $\dot{x}_i(t) = 0$ for all i, then we will call the point as a fixed point. Once the system is at the fixed point, it reaches the equilibrium state and will not move away, since the time derivative is 0 for all state variables.

Fixed points can be classified into two categories based on the behavior of the system in its vicinity. If the trajectory starts near a fixed point and converges to the fixed point, we will say the point is attracting. This point is also stable, because if we make a small perturbation away from the point, the trajectory tends to move back close to the fixed point. On the other hand, if the trajectory diverges from a fixed point nearby, the point is said to be repelling. It is a unstable fixed point as well, since a small perturbation will drive the trajectory away from the point.

Sometimes the trajectory performs a periodic behavior, which falls into the category of limit cycles. A limit cycle is defined as an isolated closed trajectory [28]. Isolated refers to the fact that there are no nearby closed trajectories. As a result, by this definition, a limit cycle will only appear in non-linear systems, even though a linear system can still have closed orbits. Similar to the fixed points, we can say a limit cycle is attracting or repelling, if all trajectories close to the limit cycle are moving towards or away from it.

Attractor is another useful concept. Usually, in a non-linear system, the trajectory will only travel in a small subset of the whole phase space. This small subset is intuitively called an attractor. As in [28], a attractor is defined to be a minimal invariant set that attracts an open set of initial conditions. Here, attracting an initial condition means that if we start from that initial condition $x(0)$, the distance between the attractor and the point on the trajectory $x(t)$ goes to 0 as $t \to \infty$. Moreover, if an attractor has sensitive dependence on the initial condition, it will be defined as a strange attractor [28], a concept closely related to chaotic systems.

2.1.2 Chaotic Systems

In many situations, in fact, probably most of the interesting cases, the system cannot be described by the motion around a fixed point. Instead, the trajectory will exhibit a structured yet aperiodic behavior. It is worth noting that this aperiodic behavior is not due to any noisy inputs, but arises from the nonlinearity of the system itself. In this case, the system will be defined as a chaotic system.

The chaotic behavior has another important property: the sensitive dependence on the initial conditions. It means that, given two initial conditions, $x_1(0)$ and $x_2(0)$, with a very small difference in the state variables, the distance between the two trajectories will grow linearly in time:

$$|x_2(t) - x_1(t)| \sim |x_2(0) - x_1(0)| e^{\lambda t} \tag{2.5}$$

To explain the role of λ here, let's imagine a infinitesimal sphere around an initial condition in the phase space. As moving along the trajectory, the sphere will change its shape to a D-dimensional ellipsoid, where D is the number of state variable in the system. There are D principle axis of the ellipsoid, which are named as Lyapunov exponents. λ in Eq. 2.5 is therefore the largest Lyapunov exponent, and it has to be positive in this case, since we know the error is growing exponentially in a chaotic system.

It is also worth pointing out that the distance between two trajectories will not grow exponentially forever. The limit of the distance is the diameter of the attractor. After it reaches the limit, the distance will saturate and stay at that level.

The sensitive dependence on the initial condition dooms our ability to predict the behavior of any chaotic system. As one can easily imagine, no matter how accurate we can estimate the initial condition, there will always be error in the estimations due to the measuring method and equipment. The error, however small it is, will grow exponentially fast and eventually cause any prediction to fail. Therefore, if we have a time window with measurements, and want to predict how the system will evolve later, all we can do is to estimate the variables and parameters as accurate enough as we can during the measurement window, so that the predictions can be relatively good for a long enough period that we care about.

2.1.3 An Simple Example: the Lorenz 63 System

The well-known Lorenz 63 system is a perfect example to illustrate the ideas in nonlinear systems that we have seen above. As the name suggested, the model is proposed by Edward Lorenz in 1963, in order to describe the atmospheric convection [17]. It contains three ordinary differential equations:

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= \sigma(y-x) \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= x(r-z)-y \\ \frac{\mathrm{d}z}{\mathrm{d}t} &= xy-bz \end{aligned} \tag{2.6}$$

where x, y and z are variables proportional to the intensity of the convective motion, the temperature difference and the distortion of the vertical temperature respectively [17]. The parameter σ is the Prandtl number, r is the Rayleigh number, and b is a dimensionless constant [27].

This system shows several interesting properties. One of these that has been studied a lot in the past is the fact that it behaves significantly differently with different parameters, and therefore could be used here to demonstrate the different behaviors of the nonlinear systems that we discussed above. For convenience, we will keep $\sigma = 10$, $b = 8/3$, and vary r for the following discussion.

When r is small but larger than 1, the system will have two stable fixed points. In Fig. 2.2, r is equal to 10, The trajectory starts from the initial condition at $(x,y,z) = (1,1,1)$, and is integrated for 40 unit time with a integration time step of 0.001 using the fourth order Runge-Kutta method. The top left figure shows the trajectory in the x-$\dot{x}$ space. It is clear from this graph that the trajectory is attracted to one of the fixed point, and the graph on the top right, which is the zoom-in trajectory of the last 10 unit time, further confirms it. The graph on the bottom shows the time evolution of the x variable. We can see that the value oscillates first, but eventually converges asymptotically to the fixed point.

In the opposite regime where r is large, the behavior of the system is quite complicated. It has been shown that there are some range of r that could result in a periodic trajectory [28]. In Fig. 2.3, we selected $r = 300$ and integrated again for 40 unit time. The trajectory seems to fall into a limit cycle after some time. To further show this, the trajectory of last 30 unit time is plotted on the top right, and it is evident that the system is already in the limit cycle after the first 10 unit time.

For some other choice of r, the system becomes chaotic. In Fig. 2.4, we have chosen

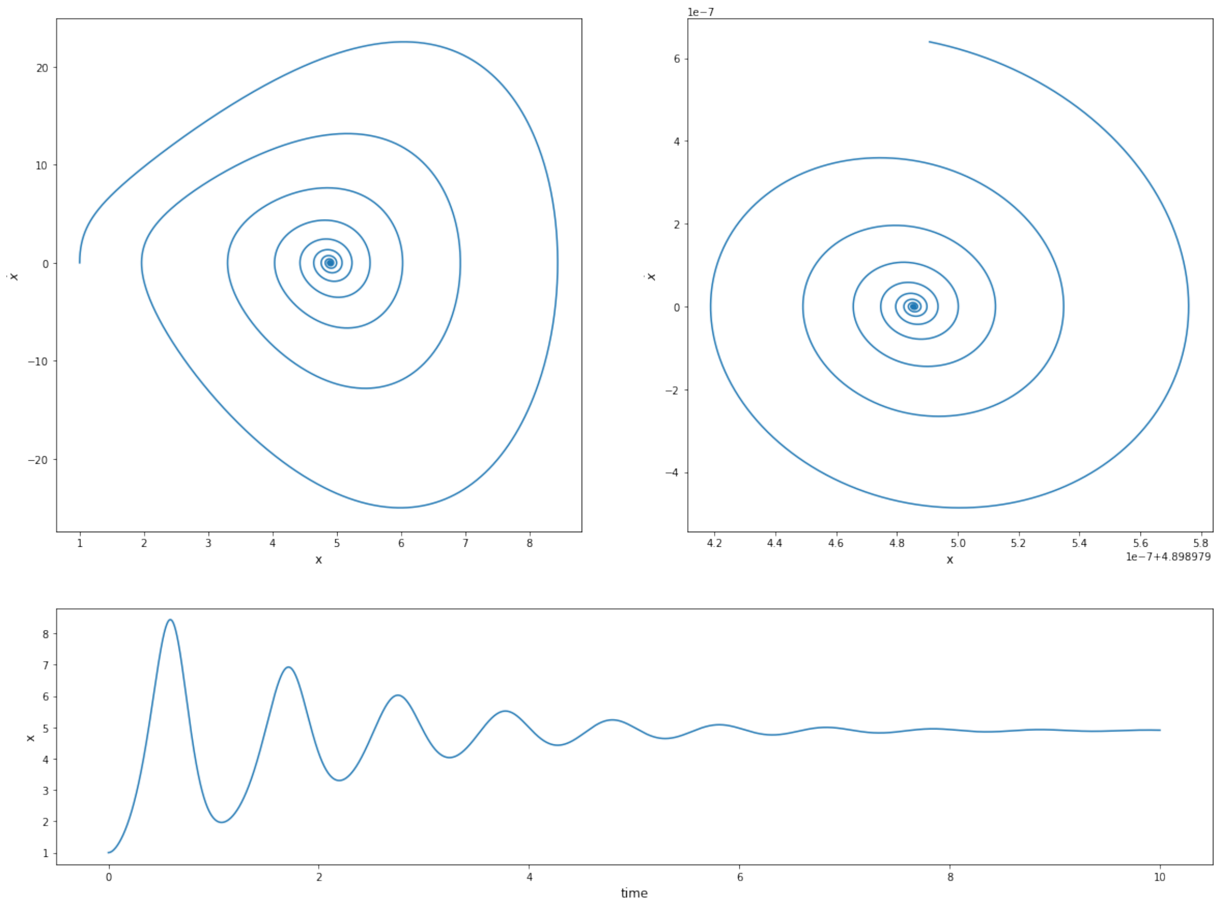


Figure 2.2: The trajectory of Lorenz 63 system.$\sigma = 10$, $b = 8/3$, $r = 10$. Top left is the trajectory in x-$\dot{x}$ space for 40 unit time. Top right is the same trajectory, but only shows for the last 10 unit time. Bottom is the graph of variable x over time. For this set of parameter, the trajectory converges to a fixed point.

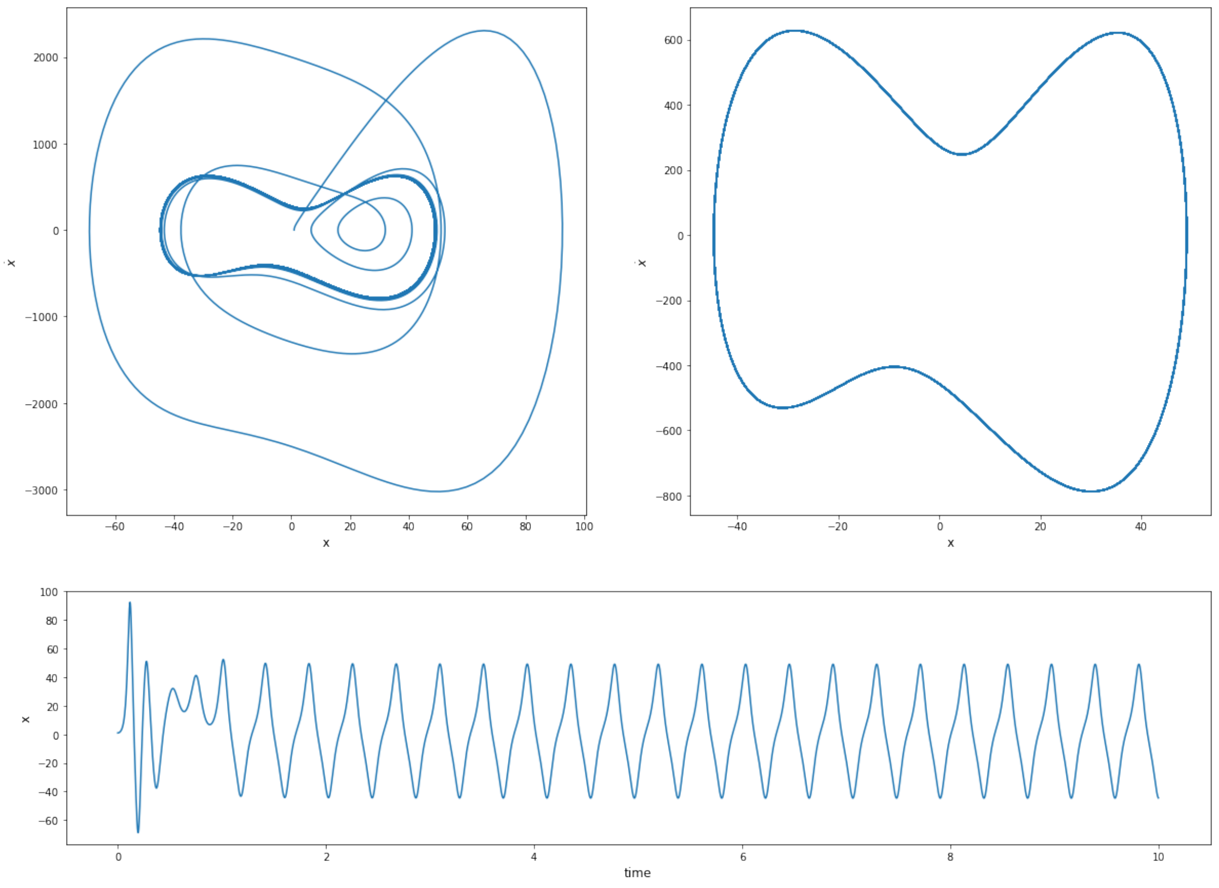


Figure 2.3: The trajectory of Lorenz 63 system.$\sigma = 10$, $b = 8/3$, $r = 300$. Top left is the trajectory in x-$\dot{x}$ space for 40 unit time. Top right is the same trajectory, but only shows for the last 30 unit time. Bottom is the graph of variable x over time. In this case, the trajectory moves in a limit cycle.

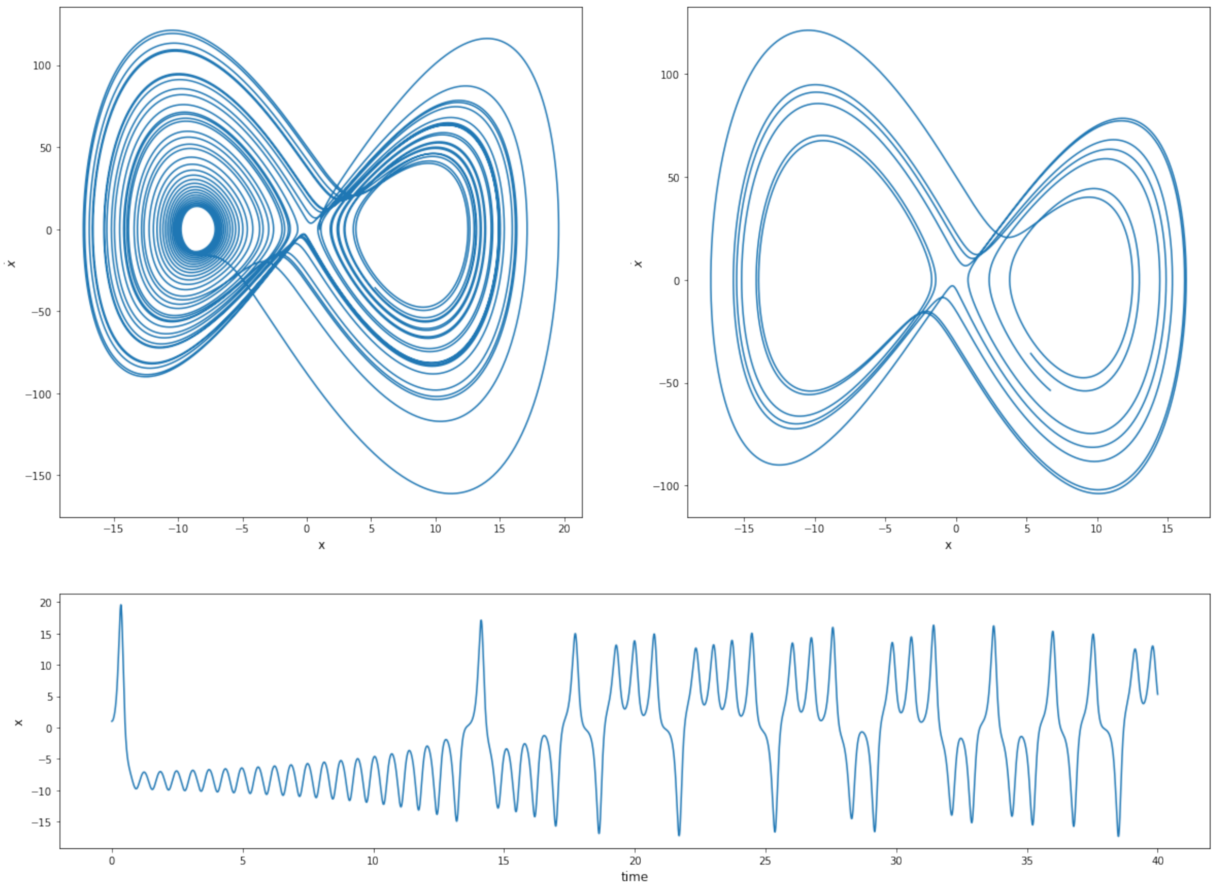


Figure 2.4: The trajectory of Lorenz 63 system.$\sigma = 10$, $b = 8/3$, $r = 28$. Top left is the trajectory in x-$\dot{x}$ space for 40 unit time. Top right is the same trajectory, but only shows for the last 10 unit time. Bottom is the graph of variable x over time. The system is chaotic for this set of parameters.

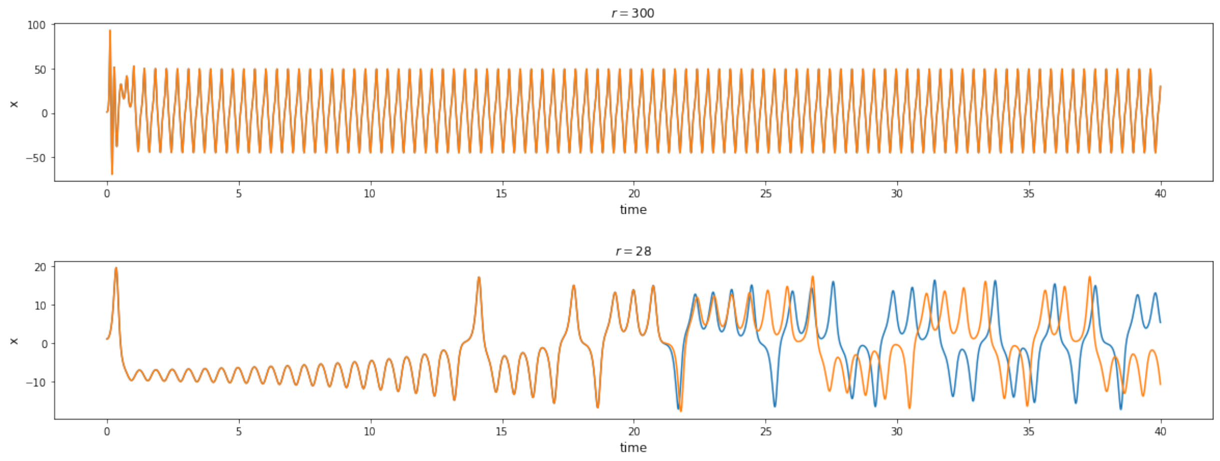


Figure 2.5: Comparison between time series of x for $r = 300$ and $r = 28$. Top: $r = 300$. Bottom 28. The two trials differ by 0.0001 in each plot.

$r = 28$, which is the value used in the original paper written by Lorenz [17]. For this set of parameters, the system is in the chaotic regime. As we can see in the top left and top right, which is the trajectory of 40 unit time and the last 10 unit time respectively, during the whole period of time that we simulated, the trajectory is highly structured and moves around two attractors, but it does not orbit in a periodic behavior, which is clear from the top right graph: the trajectory never really overlap like it does in a periodic behavior in Fig. 2.3. The x vs time graph at bottom can also be an evidence to this fact. We can see that the trajectory first takes about 13-14 unit time orbiting around the attractor in the negative x region. Afterwards, when it reaches some critical distance from the attractor, it leaves the attractor, and starts to moves between the two attractor in a chaotic behavior, as there is apparently no periodic cycle shown from 14-40 unit time.

Fig. 2.5 shows the time plot for two different initial conditions, differing by 0.0001, in both cases when $r = 300$ and $r = 28$. As expected, when the trajectory is periodic when $r = 300$, the error between two trials stays at a constant level, and the two trajectories almost overlap with each other on the plot. On the other hand, in the chaotic regime, as it is sensitively dependent on the initial condition, we can find that, after the trajectories start to move around the two attractors,

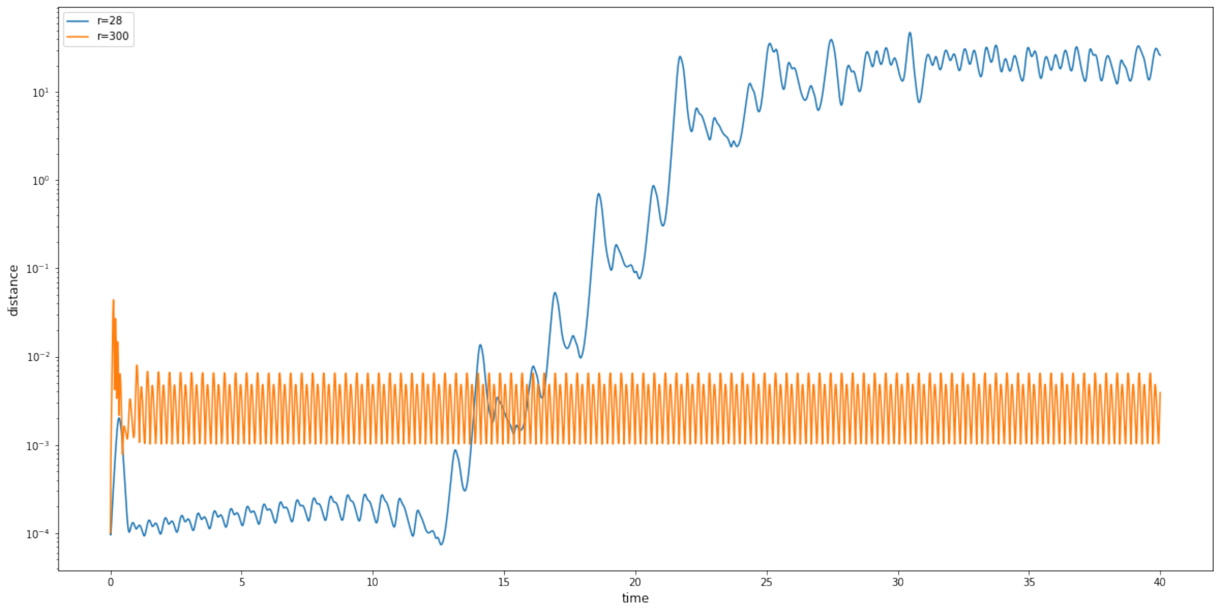


Figure 2.6: Distance between the two trajectories in Fig. 2.5 when $r = 28$ and $r = 300$. Y axis is in log scale.

the distance between the two trajectory diverges very fast and start to be distinguishable after about 20 unit time.

Fig. 2.6 plotted the distance between the two trajectories over time in both cases. When $r = 300$, it oscillates at the level around 10^{-3} to 10^{-2}, due to the periodic behavior. When $r = 28$, it is clear that the distance starts to grow rapidly from 13 unit time, which coincide with the time it leaves the initial attractor it orbiting around. Then the distance fluctuates and grows exponentially, as the expected behavior of a chaotic system, until it reaches the level close to the size of the attractor at around 10 to 50.

2.2 Probabilistic Approach

Another way of solving the problem is from probabilistic perspective. For convenience, we shall define the following notation: $\mathbf{Y}(k) \equiv \{\mathbf{y}(0), \mathbf{y}(1), \ldots, \mathbf{y}(k)\}$ as the measurement from the beginning of the measurement window to time τ_k and $\mathbf{X}(k) \equiv \{\mathbf{x}(0), \mathbf{x}(1), \dots, \mathbf{x}(k); \mathbf{p}\}$ as the set of parameters and state variables starting from t_0 to t_k. In order to transfer information from measurements to the model, we need the conditional probability distribution $P(X|Y)$.

Now, leveraging the Markov property of equations 2.2, we could write down the conditional probability $P(\mathbf{X}(k+1)|\mathbf{Y}(k+1))$ as [1]

$$P(\mathbf{X}(k+1)|\mathbf{Y}(k+1)) = \frac{P(\mathbf{x}(k+1), \mathbf{y}(k+1), \mathbf{X}(k)|\mathbf{Y}(k))}{P(\mathbf{y}(k+1)|\mathbf{Y}(k))} \cdot \frac{P(\mathbf{x}(k+1)|\mathbf{x}(k))P(\mathbf{X}(k)|\mathbf{Y}(k))}{P(\mathbf{x}(k+1), \mathbf{X}(k)|\mathbf{Y}(k))} \tag{2.7}$$

If we use Shannon's conditional mutual information, which is defined as $\mathrm{CMI}(a,b|c) \equiv \log\left[\frac{P(a,b|c)}{P(a|c)P(b|c)}\right]$ [7], then $CMI(\mathbf{y}(k+1), \mathbf{x}(k+1), \mathbf{X}(k)|\mathbf{Y}(k))$ is:

$$CMI(\mathbf{y}(k+1), \mathbf{x}(k+1), \mathbf{X}(k)|\mathbf{Y}(k)) \equiv log[\frac{P(\mathbf{x}(k+1), \mathbf{y}(k+1), \mathbf{X}(k)|\mathbf{Y}(k))}{P(\mathbf{y}(k+1)|\mathbf{Y}(k))P(\mathbf{x}(k+1), \mathbf{X}(k)|\mathbf{Y}(k))}] \tag{2.8}$$

Shannon's conditional information tells us how much information we obtain about a when b is measured conditioned on c. In this context, it shows that the information can be transferred from the measurements $\mathbf{Y}(k+1)$ to the state variables and parameters $\mathbf{x}(k+1)$.

Using Eq. 2.8 we can simplify Eq. 2.7 as

$$P(\mathbf{X}(k+1)|\mathbf{Y}(k+1)) = \exp[CMI(\mathbf{y}(k+1), \mathbf{x}(k+1), \mathbf{X}(k)|\mathbf{Y}(k))] \cdot P(\mathbf{x}(k+1)|\mathbf{x}(k))P(\mathbf{X}(k)|\mathbf{Y}(k)) \tag{2.9}$$

The recursion relation that we have derived in Eq. 2.9 can help us write down a expression for $P(\mathbf{X}|\mathbf{Y})$:

$$P(\mathbf{X}|\mathbf{Y}) \propto \left\{ \prod_{k=1}^{F} P\left(\mathbf{y}(k)\,|\mathbf{X}(k)\,,\mathbf{Y}(k-1)\right) \prod_{n=1}^{F} P(\mathbf{x}(n)|\mathbf{x}(n-1)) \right\} P(\mathbf{x}(0)) \tag{2.10}$$

Eq. 2.10 can also be written as $P(\mathbf{X}|\mathbf{Y}) \propto \exp[-A(\mathbf{X})]$, where $A(\mathbf{X})$, the negative log likelihood, is the definition of the action. Derived from Eq. 2.7,

$$\begin{aligned} A(\mathbf{X}) = -\sum_{k=1}^{F} & \left\{ \log\left[P\left(\mathbf{y}(k)\,|\mathbf{X}(k)\,,\mathbf{Y}(k-1)\right)\right] - \sum_{n=0}^{N} \log[P(\mathbf{x}(n+1)|\mathbf{x}(n))] \right\} \\ & - \log[P(\mathbf{x}(0))] \end{aligned} \tag{2.11}$$

To simplify the expression for $A(\mathbf{X})$ in Eq. 2.11, we will assume that the noise in the model and measurements are Gaussian noise with zero mean and diagonal precision matrix R_f and R_m respectively. Then Eq. 2.11 can be further written as

$$A(\mathbf{X}) = \sum_{k=1}^{F}\sum_{l=1}^{L} \frac{R_m(l)}{2} \left(x_l\,(k) - y_l\,(k)\right)^2 + \sum_{n=0}^{N-1}\sum_{a=1}^{D} \frac{R_f(a)}{2} \left(x_a(n+1) - f_a(\mathbf{x}(n), \mathbf{p})\right)^2 \tag{2.12}$$

where $R_m(l)$ represents the lth diagonal elements of the precision matrix R_m, and $R_f(a)$ represents the ath diagonal elements of R_f. The two terms will be referred to as measurement error and model error respectively. Here, the term $-\log[P(\mathbf{x}(0))]$ is omitted because it actually does not affect any result, as we will see soon.

Suppose we now have a function of interest $G(\mathbf{X})$. Its expected value along the path $\mathbf{X}$ given measurements $\mathbf{Y}$ is

$$E[G(\mathbf{X})|\mathbf{Y}] = \langle G(\mathbf{X})\rangle = \frac{\int d\mathbf{X} G(\mathbf{X}) e^{-A(\mathbf{X})}}{\int d\mathbf{X} e^{-A(\mathbf{X})}} \tag{2.13}$$

where $d\mathbf{X} = \prod_{n=0}^{N} d^D\mathbf{x}(n)$ and all other parts in the action cancel out in the denominator and numerator. If we replace $G(\mathbf{X})$ with $x_a(n)$ or a single parameter p, then Eq. 2.11 will give us the expected value of the state variable or the parameter, which is important in data assimilation.

2.3 Evaluating the Path Integral

Now our goal becomes evaluating the path integral in Eq. 2.13. There are a few challenges in doing that. One of them is the large number of dimensions in $\mathbf{X}$ space. For example, in Lorenz 96 model that we typically used to test our methods, a normal setup we will choose is 20 variables with 200 time steps. Then the number of dimensions in $\mathbf{X}$ space is 4000. In a model that describes the underlying dynamics more accurately but also more complicated, the number is usually several order of magnitude higher.

Another challenge is the non-linearity of $A(\mathbf{X})$. While both terms in $A(\mathbf{X})$ are quadratic, the model $f_a(\mathbf{x}(n), \mathbf{p})$ is non-linear in most interesting problems, and therefore the integral in Eq. 2.13 is not Gaussian.

In general, there are two useful methods to evaluate this integral, which is high dimensional and non-linear: Laplace's method [14] and Monte Carlo sampling [13] [22] [34].

The Laplace's method states that

$$\int_a^b h(x)e^{Mg(x)}dx \approx \sqrt{\frac{2\pi}{M\left|g''\left(x_0\right)\right|}}h\left(x_0\right)e^{Mg\left(x_0\right)} \text{ as } M \to \infty \tag{2.14}$$

where x_0 is the global minimum of function $g(x)$.

Applying the approximation to the integral 2.13 and canceling out the identical terms, we can find $\langle G(\mathbf{X})\rangle \approx G(\mathbf{X}_0)$, where $G(\mathbf{X}_0)$ corresponds to the path that gives the global minimum of the action.

Laplace's method turns the integral into a problem of finding the global minimum, which can be solved by some well-developed optimization routine such as L-BFGS [16], but it has several drawbacks.

The first disadvantage is that most optimization methods require calculating or approximating Hessian matrix, which needs a large number of computing time and memory space. The amount of memory and computing time required is reasonable for a problem that has 4000-dimension variable space, like the Lorenz 96 model I mentioned before. When the number of dimension grows larger, the requirement will increase drastically and become unrealistic at larger scales.

Another drawback is that, while we find the global minimum of $A(\mathbf{X})$ as the dominant contribution to the integral, the conditional probability distribution around the minimum is not sampled. If we want to evaluate the corrections to this dominating contribution, the perturbation methods are in need. the convergence rate of such methods, however, is very sensitive to the form of $A(\mathbf{X})$.

Monte Carlo sampling provides another way to evaluate the integral 2.13, which has been utilized for decades [19] [8] [22]. The well-developed Metropolis-Hastings Monte Carlo method [19] [8] can estimate the conditional probability distribution $P(\mathbf{X}|\mathbf{Y})$ by starting from a random place in $\mathbf{X}$ space, making random walk proposals, and accepting or rejecting the proposals with a probability based on detailed balance:

$$\pi\left(s'\right)P\left(s',s\right) = \pi(s)P\left(s,s'\right) \tag{2.15}$$

Here, π is a stationary distribution, and $P\left(s,s'\right)$ is the transition probability from state s to state s'. This detailed balance condition assures the Markov chain to be reversible. After a sufficient number of proposals, we will have a good estimation on the expected value $\langle G(\mathbf{X})\rangle$, and the corresponding error will be proportional to the inverse square root of the number of proposals.

2.4 Data Assimilation Methods

In this section, I will talk about a few well established and widely used data assimilation methods, including two of the mostly discussed methods: the Ensemble Kalman Filter, and the 4-Dimensional Variational method.

2.4.1 The Nudging Method

In order to achieve the synchronization between the model and the measurements, an intuitive way would be to modify our model by adding a term that couples the estimated variables with the observed data. This is the basic idea of a commonly used method, nudging or Newtonian relaxation [10]. Using the nudging method, the equation of the model will be modified as

$$x_a(n+1) = f_a(\mathbf{x}(n),\mathbf{p}) + \delta_{al}\delta_{kn}g(t)(y_l(k) - x_a(n)) \tag{2.16}$$

l denotes the index of measured L variables. The two Kronecker delta indicates that we only apply nudging when there is a measurement. $g(t)$ is the nudging coefficient to control the strength of the nudging terms, which is usually chosen empirically. If it is too small, then the error in the estimates would become too large before the nudging is effective. On the other hand, if $g(t)$ is too large, then the estimates would be driven too rapidly to the measurement, and the model may not have time to adjust [12].

If there is enough measurements, then the synchronization between the model and the measurements will be achieved in all dimensions, despite being observed or not, under the effects of nudging. Fig. 2.7 is a demonstration of the nudging results on Lorenz 96 model [18], a instructional model in meteorology that I will discuss more in detail in Chapter 4. We can see that all the variables shown are synchronized to the measurements, even for the unobserved variables, where the initial guesses are far from the actual values.

2.4.2 Ensemble Kalman Filter

Ensemble Kalman Filter, first proposed by Evensen [5], is a widely-used data assimilation methods in the study of meteorology. It is a Monte Carlo approach developed based on the Extended Kalman Filter, which is easier to implement and more computationally efficient.

We will still write the system in the way we wrote Eq. 2.3 and 2.4, but specify the error in each time step here:

$$\begin{aligned}\mathbf{x}(n) &= \mathbf{F}(\mathbf{x}(n-1)) + \eta_n \\ \mathbf{y}(n) &= \mathbf{H}\mathbf{x}(n) + \varepsilon_n\end{aligned} \tag{2.17}$$

Both the dynamical equation and the equation for measurements above are written in the vector form. Therefore, the matrix form $\mathbf{F}$ is used to represent the dynamics of the system. n

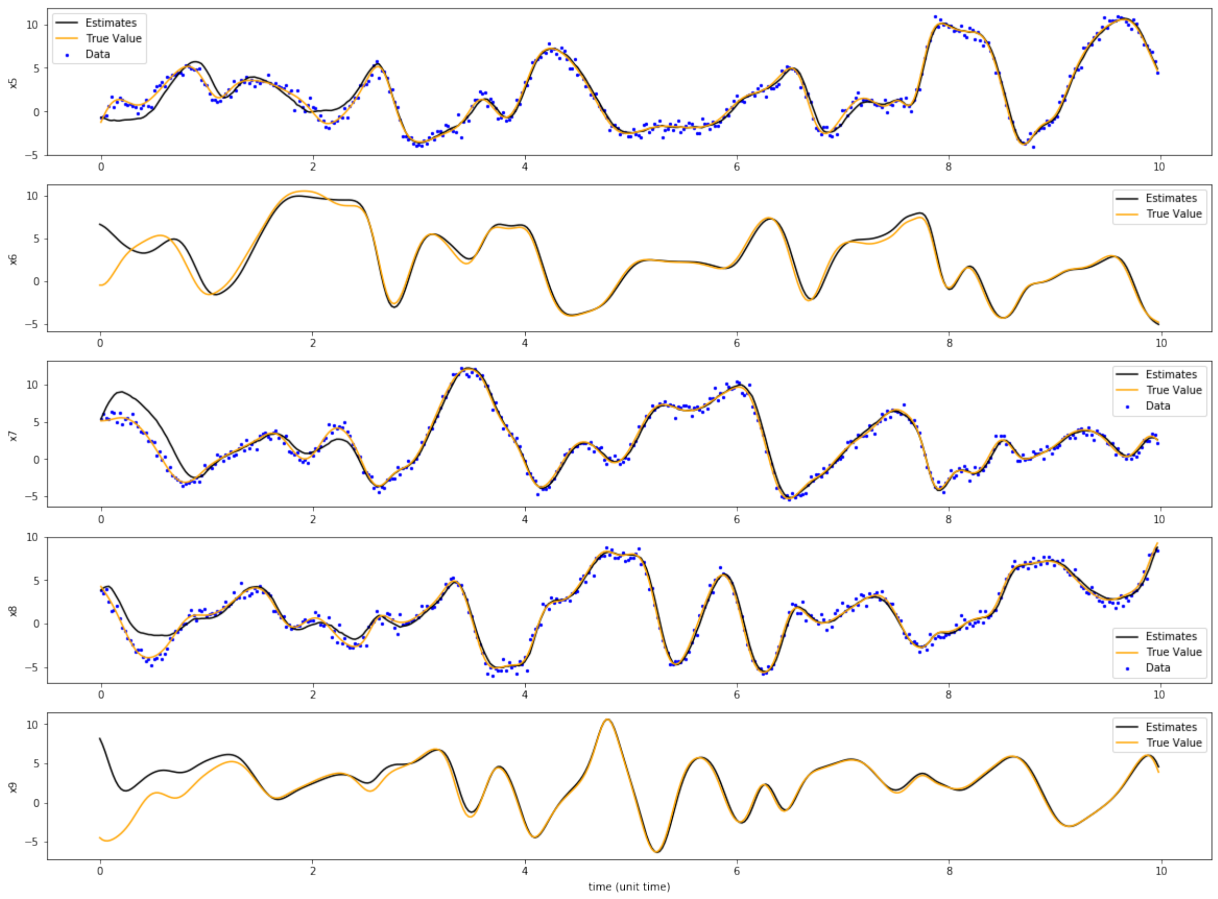


Figure 2.7: Nudging results on Lorenz 96 model, with 14 out of 20 variables measured. Black lines are the estimated paths, yellow lines are the actual paths, and blue dots are the measurements with noise on the actual paths. x_6 and x_9 are unobserved variables, while the other three are observed. It is clear that the synchronization is achieved on all variables at the end of measurement window.

denotes the nth time step. η_n and ε_n denote the error or noise in the measurements and model respectively at time step n.

The traditional Kalman Filter involves two steps: predicting the current state variables and covariance matrix based on the past analysis, and updating the current state variables and covariance matrix following the analysis scheme, which will be described soon. In this section, I will use the similar notation as in [6]. $\mathbf{x}^f$ and $\mathbf{C}^f$ are the variables and covariance matrix of the predicted or first guess values.$\mathbf{x}^a$ and $\mathbf{C}^a$ are the corresponding updated or analyzed values. Then the prediction step can be formulated as

$$\begin{aligned} \mathbf{x}^f(n) &= \mathbf{F}(\mathbf{x}^a(n-1)) \\ \mathbf{C}_x^f(n) &= \mathbf{F}\mathbf{C}_x^a(n-1)\mathbf{F}^T + \mathbf{C}_\varepsilon(n-1) \end{aligned} \tag{2.18}$$

where $\mathbf{C}_x(n)$ and $\mathbf{C}_\varepsilon(n)$ are the covariance matrix of the variables and error in model respectively at time step n.

After the prediction step, we need to update the variables and covariance matrix:

$$\begin{aligned} \mathbf{K} &= \mathbf{C}_x^f(n)\mathbf{H}^T(\mathbf{H}\mathbf{C}_x^f(n)\mathbf{H}^T + \mathbf{C}_\varepsilon(n))^{-1} \\ \mathbf{x}^a(n) &= \mathbf{x}^f(n) + \mathbf{K}(\mathbf{y}(n) - \mathbf{H}\mathbf{x}^f(n)) \\ \mathbf{C}_x^a(n) &= (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{C}_x^f(n) \end{aligned} \tag{2.19}$$

In Eq. 2.19, $\mathbf{I}$ is the identity matrix, and matrix $\mathbf{K}$ is called the Kalman gain. This regular Kalman Filter method, nevertheless, only works for linear models. In order to apply to the nonlinear systems, we need to use the extended Kalman Filter.

If we denote the Jacobian of $\mathbf{F}$ at time t as $\mathbf{F}'_t$, neglecting higher order terms, the extended Kalman Filter will just have a different prediction step, which can be written as

$$\begin{aligned} \mathbf{x}^f(n) &= \mathbf{F}(\mathbf{x}^a(n-1)) \\ \mathbf{C}_x^f(n) &= \mathbf{F}'_{n-1}\mathbf{C}_x^a(n-1)\mathbf{F}'^T_{n-1} + \mathbf{C}_\varepsilon(n-1) \end{aligned} \tag{2.20}$$

The implementation of Extended Kalman Filter has a few disadvantages that people have encountered. One of them is the huge amount of computation time and space required to do the integration of the covariance matrix related to the Jacobian of the model, which is a large matrix multiplication in a high-dimensional space. The Ensemble Kalman Filter method is thus developed to solve this issue.

In the Ensemble Kalman Filter method, we need to first form an ensemble of measurement with N members.

$$\mathbf{y}_j = \mathbf{y} + \varepsilon_j \tag{2.21}$$

where $j = 1, 2, ..., N$, and ε_j is random noise with mean equal to 0 and covariance equal to $\mathbf{C}_\varepsilon$. The ensemble covariance matrix for the measurement noise therefore is

$$\mathbf{C}_\varepsilon^e = \overline{\varepsilon\varepsilon^T} \tag{2.22}$$

The prediction step of $\mathbf{x}^f$ is still the same for each ensemble member. Then, instead of calculating the covaraince matrix $\mathbf{C}_x^f(n)$ at each time, we use an ensemble estimation:

$$\mathbf{C}_x^{ef} = \overline{(\mathbf{x}^f - \overline{\mathbf{x}^f})(\mathbf{x}^f - \overline{\mathbf{x}^f})^T} \tag{2.23}$$

Now, the update steps are

$$\begin{aligned}
\mathbf{K}^e &= \mathbf{C}_x^{ef}\mathbf{H}^T(\mathbf{H}\mathbf{C}_x^{ef}\mathbf{H}^T + \mathbf{C}_\varepsilon^e)^{-1} \\
\overline{\mathbf{x}^a} &= \overline{\mathbf{x}^f} + \mathbf{K}^e(\overline{\mathbf{y}} - \mathbf{H}\overline{\mathbf{x}^f}) \qquad (2.24) \\
\mathbf{C}_x^{ea} &= (\mathbf{I} - \mathbf{K}^e\mathbf{H})\mathbf{C}_x^{ef}
\end{aligned}$$

In the equations above, the time label is omitted for simplicity, since in the analysis step, the time step is the same for all the matrices and vectors.

2.4.3 4-Dimensional Variational (4D-VAR) Method

In the previous sections, I have introduced several sequential data assimilation methods that are commonly used. Now, I would like to discuss about a variational method, the 4D-VAR method.

The 4D-VAR data assimilation has been applied extensively in the study of meteorology [21]. In general, it seeks to optimize a cost function which represents the error in the measurements and the model.

In the formulation of 4D-VAR, the assumption of Gaussian noise is assumed. Therefore, the cost function is similar to Eq. 2.12:

$$A(\mathbf{X}) \propto R_b|\mathbf{x}_0 - \mathbf{x}_b|^2 + \sum_{n=0}^{N} R_m|\mathbf{H}\mathbf{x}(n) - \mathbf{y}|^2 + \sum_{n=0}^{N-1} R_f|\mathbf{x}(n+1) - \mathbf{F}\mathbf{x}(n))|^2 \qquad (2.25)$$

In the equation above, there is one more term compared to Eq. 2.12: $R_b|\mathbf{x}_0 - \mathbf{x}_b|^2$. This term, the 'background' term, represents the deviation from the estimated initial condition $\mathbf{x}_b$.

Now that we have a cost function, what is left is to minimize it with respect to $\mathbf{x}(n)$, for

$n = 0, 1, ..., N$, and there are a variety of methods for this purpose. For example, the conjugate gradient descent with Lanzcos algorithm is used in [21].

The use of Eq. 2.25 without any further assumption is usually referred to as the weak constrained 4D-VAR method. the term ”weak” is simply stating the fact that we allow our model to have error. In other words, we have a finite value of R_f. However, this approach cannot be applied to high dimensional problems, as the memory required is $O((N+1)D)$, which makes it impractical for some real applications, such as weather forecasts where D can be over 10^6 [12]. In such cases, a further simplification to Eq. 2.25 can be used. If we assume that the model is almost perfect, then in the limit of $R_f \to \infty$, the cost function becomes

$$A(\mathbf{X}) \propto R_b|\mathbf{x}_0 - \mathbf{x}_b|^2 + \sum_{n=0}^{N} R_m|\mathbf{H}\mathbf{x}(n) - \mathbf{y}|^2 \tag{2.26}$$

under the constraint $\mathbf{x}(n+1) = \mathbf{F}\mathbf{x}(n)$.

In this formulation, the cost function is then only a function of the initial condition, which reduces the memory usage to $O(D)$. This method is called the strong constrained 4D-VAR method, as it makes an assumption that the model is perfect, thus acts as another set of constraints to the cost function.

Even though we can barely propose any perfect model, the strong constrained 4D-VAR method usually performs reasonably well, and therefore has an extensive use in the study of meteorology [21] [24].

2.5 Twin Experiments

While we are developing methods and algorithms for experimental data, it is usually not effective to test and improve the methods themselves using real experimental data. The reason is that the real world is usually much more complicated than we could describe using models, and there are always unknown underlying dynamics that are not accounted by models. If we use real data and our technique fails, we are not able to know if it is due to the technique we use, or from the wrong or incomplete model. Therefore, an idea called twin experiments can be used to test our data assimilation methods [1].

To do twin experiments, we will first generate a true path by integrating forward using the underlying dynamical model, Eq. 2.1. By tuning the initial conditions and parameters in the model, we can conveniently generate a true path in the regime that we are interested in. Then random noise, usually Gaussian, will be added to the true path as our measurements. Lastly, we will use part of these ”measurements” to test our data assimilation methods as the available incomplete and noisy data.

The twin experiments will not only test the effectiveness of our methods, it will also provides important information of the model itself, like how many measurements are necessary in order to make good predictions. After validating the model using the twin experiments, it is then ready to work on the real experimental data, which is generally much more complicated. This workflow from the twin experiments to real experiments has been used and succeeded [20].

acknowledgements

Chapter 2, in part, has been adapted from Z. Fang, A. S. Wong, K. Hao, A. J. Ty, and H. D. Abarbanel. Precision annealing monte carlo methods for statistical data assimilation and machine learning. *Physical Review Research*, 2(1):013050, 2020 and A. S. Wong, K. Hao, Z. Fang, and H. D. Abarbanel. Precision Annealing Monte Carlo Methods for Statistical Data Assimilation: Metropolis-Hastings Procedures. arXiv preprint arXiv:1901.04598 (2019). The author of this thesis was a co-first author of these papers.

Chapter 3

Precision Annealing Method

In this chapter, I will discuss about the precision annealing method. The idea is to do annealing on R_f of the model error term in Eq. 2.12. As we have mentioned in Chapter 2, either Laplace's method or Monte Carlo sampling can be combined with precision annealing. I will first briefly talk about variational annealing, which uses Laplace's method. Variational annealing is first proposed in [23] and has succeeded in several different problems [31] [32]. Then I will focus on Monte Carlo precision annealing (PAMC) method, which, as the name suggests, uses Monte Carlo sampling instead of Laplace's method.

3.1 Variational Annealing

For the problem setup we have introduced in the previous chapter, it is shown that the expected value of a function $\langle G(\mathbf{X}) \rangle$ can be approximated as $G(\mathbf{X}_0)$ where $\mathbf{X}_0$ is the path in which the action $A(\mathbf{X})$ defined in Eq. 2.12 takes its global minimum. Now the problem is converted to an optimization problem, where the global minimum needs to be found, but the non-linear model $f_a(\mathbf{x}(n), \mathbf{p})$ makes this task very difficult, as it may contain several or many local minima, and therefore is hard for any optimization algorithm to find the global minimum.

Variational annealing is a method that would alleviate this issue. By starting from a small value of R_f, the measurement error will dominant in the action. Since the measurement error is quadratic, the optimization algorithm can easily find the global minimum near the measured values. However, as we have a small coefficient in the model error, the square error $(x_a(n+1) - f_a(\mathbf{x}(n), \mathbf{p}))^2$ may be relatively large, meaning that the estimated path does not follow the dynamical model well.

Then, the value of R_f will be adiabatically increased, while the optimization routine keeps tracking the minimum of the action. Eventually, when R_f is large enough, for instance, a few order of magnitude larger than R_m, the model error will dominant in the action, and the minimum we arrive at should contain a very small model error, which means that the path is now strictly following the model within the tolerance.

The whole procedure of variational annealing can be summarized as below:

1. Initialize the path $\mathbf{X}_0$ in the following way: For the measured variable x_l, set the value to the measurement, $x_l(t) = y_l(t)$. For the unmeasured variable, initialize its value randomly from a uniform distribution within the dynamical range of the variable. The action $A(\mathbf{X}_0)$ on this initial path is calculated.

2. Starting from a small value $R_f = R_{f0}$, the optimization procedure is performed on the initial path $\mathbf{X}_0$, to minimize the action. Typical values we use for R_{f0} are usually 2-3 order of magnitude smaller than the R_m value. We will call the resulting path as $\mathbf{X}_1$, and the corresponding action $A(\mathbf{X}_1)$ is recorded.

3. The value of R_f is then increased from R_{f0} to $R_{f0}\alpha$, where $\alpha > 1$ is a constant. The optimization procedure is again performed on $\mathbf{X}_1$, leading to a new path $\mathbf{X}_2$, with the minimized action $A(\mathbf{X}_2)$.

4. R_f is increased in the same manner until to $R_{f0}\alpha^{\beta}$ for a large enough β. Each path $\mathbf{X}_{\beta}$ and its action $A(\mathbf{X}_{\beta})$ are stored. Empirically, the final value of R_f is several order of magnitude larger than R_m, to ensure we find a solution with small model error.

We shall note that, the variational annealing method inherits all the disadvantages of the Laplace's method we have mentioned in Chapter 2. As a result, it is not suitable for problems with large number of dimensions, which is the motivation of developing its Monte Carlo alternatives, the precision annealing Monte Carlo method.

3.2 Precision Annealing Monte Carlo (PAMC)

As shown in Chapter 2, Monte Carlo method could replace Laplace's method in evaluating the path integral, which is the basic idea of PAMC. The procedure of PAMC is similar to the variational annealing, but I will go into more details at each step.

3.2.1 Initialization of Paths for PAMC

When the path is initialized, we would like it to represent the minimum of action at $R_f = 0$, since that is the actual starting point of the annealing procedure. When $R_f = 0$, the only term in the action is the measurement error. As a result, the path $\mathbf{X}_0$ to minimize the action need to satisfy $x_l(t) = y_l(t)$ for a measured variable x_l. For unmeasured variables, because they do not appear in the measurement error, the action level is degenerate in those directions and any choice would minimize the action. This is the reasoning behind the initialization of variational annealing. In PAMC we would slightly revise this initialization process.

Even though we can start anywhere using the Monte Carlo method and eventually it will give a good estimate on the expected value, the time it takes may be undesirably long if we are unfortunate and start from a position far away from the dominating peak in the distribution. Initializing the observed variables to the measurements will guarantee that we start from a position close to the region that will contribute most to the path integral. Random choices of unobserved variables, nevertheless, are usually not the best options. Hence in practice, we will use the following way to initialize the path.

First, we still choose randomly from a uniform distribution at the beginning of the measurement window, construct a D-dimensional vector in the way we have described above, where D is the number of state variables in the model. We denote this vector as $\mathbf{x}(0)$.

Then, we integrate the model forward as $f(\mathbf{x}(0),\mathbf{p})$ and replace the measured variables with $y_l(1)$ to get the state variables $\mathbf{x}(1)$. The same step is repeated: integrate forward for one time step, and replace observed variables with measurements, until the end of the time window. These vectors $\mathbf{x}(0),\mathbf{x}(1),\ldots,\mathbf{x}(n)$ together form the initial path $\mathbf{X}_0$.

Instead of randomly choosing the unobserved variables, the above method constructs a path that has 0 model error for those unobserved variables even if R_f is not 0, and thus would have a higher probability to be close to the dominating part of the path integral.

Another thing worth noting is that, for both variational annealing and PAMC in practice, we would choose N_I different initial paths rather than just one, which will be written as $X_0^{(i)}$, for $i = 0,1,\ldots,N_I$. The reason for this will be explained later in this chapter.

3.2.2 $R_f = R_{f0}$, the First Step Using Monte Carlo Method

Now we begin with a small R_{f0}. In PAMC, practically we use a larger R_{f0} value than that is used in variational annealing, usually about $0.1R_m$. There are two reason for this choice. First, we have already initialized the path in a way to minimize the action. So the initial path $\mathbf{X}_0$ ideally should be close to the final path at large R_f. Thus a very small R_{f0} to reduce the non-linearity is not necessary. Second, because the Metropolis-Hastings Monte Carlo algorithm uses random proposal to choose a new path, if R_{f0} is very small, in the direction of unobserved variables, there is a decent probability that the proposal will wander off from the region around the dominating peak, and therefore lead to difficulty in estimating the path integral when we increase the R_f value later.

At this point, we will use the Metropolis-Hastings procedure to propose new paths. The changes Δx from the old path to the new path are chosen uniformly from $[-\delta_a, \delta_a]$, where δ_a is the random proposal step size for state variable x_a. To achieve a higher acceptance rate and therefore faster run time, the proposed changes are made one variable at a time. After the change is proposed, namely to replace x_a with $x'_a = x_a + \Delta x_a$, the new action is calculated and so does the difference in action $dA = A(\mathbf{X}_{new}) - A(\mathbf{X}_{old})$. This change is then accepted with a probability $P = min(1, exp(-dA))$. If it is accepted, the variable x_a will be updated to x'_a. Otherwise it will stay as x_a. After all components in path $\mathbf{X}_0$ are updated, we arrive at a new path, which shall be labeled as $\mathbf{X}_{1,1}$.

The same procedure will be repeated until we have N_{it} different new paths, $\mathbf{X}_{1,i}$ for $i = 0, 1, \ldots, N_{it}$. The value of N_{it} is chosen empirically for different problems, so that we have enough paths to estimate the expected value, and not too many to result in unnecessarily long computing time. Then the path $\mathbf{X}_1$ to start the next step of annealing procedure is calculated as

$$\mathbf{X}_1 = \frac{1}{N_{it}} \sum_{i=0}^{N_{it}} \mathbf{X}_{1,i} \tag{3.1}$$

The step size for random proposal δ_a is also a crucial choice in our Monte Carlo procedure. If it is too large, then the acceptance rate will be very low near the global minimum of the action, and results in overly high contribution from the path near this region. If it is too small, the number of paths N_{it} needed will be very high and cause a longer computing time. A solution to this issue is described in [23]. The idea is to add a initialization phase before the N_{it} proposals. During this time, the updated paths will not be counted into the calculation of $\mathbf{X}$ in Eq. 3.1. The paths will be divided into smaller blocks during the initialization phase. Within each block, the acceptance rate f_{accept} for each variable is calculated, and the step size will be updated at the end of the block using the following update rule.

$$\delta_a \leftarrow \delta_a(1 + \gamma(f_{accept} - f_0)) \tag{3.2}$$

γ is a constant to control how drastic we want to adjust the step size, and f_0 is the ideal acceptance rate we aim for, which is usually set to be 0.5. If the ideal acceptance rate is reached, at the end of initialization phase the change in step size should be small.

3.2.3 $R_f = R_{f0}\alpha^{\beta}$, Continuing the Annealing Procedure

After we find $\mathbf{X}_1^{(i)}$ for each of the distinct initial paths, the same Monte Carlo procedure is applied after we increase R_f to $R_{f0}\alpha$, and we will get $\mathbf{X}_2$ after another N_{it} updates. we continue in this manner, until β is large enough. The final average path we will use is then

$$\mathbf{X}_{\beta} = \frac{1}{N_{it}} \sum_{i=0}^{N_{it}} \mathbf{X}_{\beta-1,i} \tag{3.3}$$

For any quantity $G(\mathbf{X})$ of out interest, we can find its expected value in the same way

$$\langle G(\mathbf{X})\rangle = \frac{1}{N_{it}}\sum_{i=0}^{N_{it}} G(\mathbf{X}_{\beta-1,i}) \tag{3.4}$$

3.2.4 Change in Action as R_f Increases

A natural question to ask is, how we will know that β is large enough. Some evidence can be found in the change in action. At the beginning, since we do not force the model strictly, the model error is comparable to the measurement error or even larger than the measurement error. Later when we increase the value of R_f, the model error starts to play a more and more important role in the action. The paths that contribute most to the path integral will then be those near the global minimum of the action, where the model error is zero. As a result, while the model error decrease as increasing R_f, the total action will increase and asymptotically reach a constant value.

The expected asymptotic value that is independent of R_f can be calculated. At large R_f, as the model error essentially goes to zero, the remaining part in the action is then the measurement error

$$A(\mathbf{X}) = \sum_{k=1}^{F}\sum_{l=1}^{L}\frac{R_m(l)}{2}\left(x_l(k)-y_l(k)\right)^2 \tag{3.5}$$

Because R_m is set to be the precision matrix of the measurements, it will follow a χ^2-distribution. Therefore, the expected value of the action will be

$$\langle A(\mathbf{X})\rangle = \frac{1}{2}FL \tag{3.6}$$

However, if the number of measurements L is not enough, not all initial paths will find the global minimum. Some may actually sampling around a local minimum and is stuck there. Then during the transition from the situation where we have enough measurements to where we do not, we will see the actions from N_I different initial conditions start to split, and reach different asymptotic level. In some other cases, some of those trials may not even find a local minimum with zero model error. Then those actions will grow exponentially as β increases. Both of these behaviors indicate that we need more measurements to perform better estimations and predictions.

3.2.5 Summary of the PAMC Procedure

To summarize, the PAMC method works as follows:

1. Initialize N_I different initial paths $\mathbf{X}_0^{(i)}$ for $i = 0, 1, \ldots, N_I$, using method described in section 3.2.1.

2. Starting from a small value $R_f = R_{f0}$, after the initialization phase to determine a good step size δ_a for each variables x_a, a proposal to change one variable x_a at a time by δx is made, where δx is drawn from a uniform distribution $[-\delta_a, \delta_a]$.

3. The change in action dA is evaluated, and the proposal to update x_a is accepted with probability $P_{accept} = min(1, exp(-dA))$.

4. After all variables at all time steps are updated, we get a new path $\mathbf{X}_{1,1}^{(i)}$.

5. Repeat steps 2-4 N_{it} times, we will find N_{it} different paths $\mathbf{X}_{1,j}^{(i)}$, for $j = 0, 1, \ldots, N_{it}$. The average paths $\mathbf{X}_1^{(i)}$ for the next annealing step are then calculated as the average of the N_{it} paths.

6. Increase R_f from R_{f0} to $R_{f0}\alpha$. Starting from $\mathbf{X}_1^{(i)}$, repeat steps 2-5, and find the average paths $\mathbf{X}_2^{(i)}$.

7. Continue in the same manner as increasing R_f to $R_{f0}\alpha^{\beta}$.

8. At the last annealing step, which we will note as β_{max}, we found the average paths $\mathbf{X}^{(i)}_{\beta_{max}}$, which are our final estimates on the expected paths. Any expected value $\langle G(\mathbf{X}) \rangle$ of our interests can then be calculated.

3.3 Improve PAMC Using Hamiltonian Monte Carlo

Since the PAMC method uses the Metropolis-Hastings algorithm, which essentially makes random walk proposals, it also inherits the drawbacks of random proposals. This procedure is ergodic in theory, which means that, if a sufficient number of steps is given, the path following the proposals we have made can reach any region. Nevertheless, the convergence is usually slow in problems containing high dimensional $\mathbf{X}$ [2]. One of the reason is that, when making random proposals, the perturbations is chosen randomly from any possible magnitude and direction uniformly. As a result, the acceptance rate will be very low in high-dimensional problems.

3.3.1 Introduction to HMC

One solution to the issue described above is to use Hamiltonian Monte Carlo (HMC) methods, which takes a different path from the traditional Metropolis-Hastings Monte Carlo methods. HMC adds a set of variables $\mathbf{P}$ to the path $\mathbf{X}$, which preserves some invariants in the $\{\mathbf{X}, \mathbf{P}\}$ space. By introducing $\mathbf{P}$ and a invariant $U(\mathbf{X}, \mathbf{P})$, we are able to make large moves in $\mathbf{X}$ space with high probability of acceptance, as staying on the surface $U(\mathbf{X}, \mathbf{P}) = const$. In this case, we can sample $exp(-A(\mathbf{X}))$ much more efficiently than random proposal MC.

As [4] suggests, an intuitive way of choosing $\mathbf{P}$ is to select a set of canonically conjugate variables. Then the proposals can be made by canonical transformation preserving the quantity $H(\mathbf{X}, \mathbf{P})$, where

$$H(\mathbf{X},\mathbf{P}) = A(\mathbf{X}) + h(\mathbf{P}) \tag{3.7}$$

This is a problem very familiar to physicist for centuries, as $H(\mathbf{X},\mathbf{P})$ is simply the Hamiltonian and $\mathbf{P}$ is the canonical momentum of $\mathbf{X}$. Then we can just use the rules learnt from classical mechanics to move in $H(\mathbf{X},\mathbf{P})$ space.

If we label the movement of $\mathbf{X}$ and $\mathbf{P}$ by a scalar s, which plays the role of time in classical mechanics, then the proposals should be made following Hamilton's equations:

$$\frac{d\mathbf{X}(s)}{ds} = \frac{\partial H(\mathbf{X}(s),\mathbf{P}(s))}{\partial \mathbf{P}(s)}; \quad \frac{d\mathbf{P}(s)}{ds} = -\frac{\partial H(\mathbf{X}(s),\mathbf{P}(s))}{\partial \mathbf{X}(s)} \tag{3.8}$$

We can now propose movements following Eq. 3.8 with an appropriate s. After enough proposals are made, we can calculate the expected value of any variable or parameter the same way as in the PAMC method.

3.3.2 Implementation of HMC

In the HMC procedure discussed above, we have introduced the canonical momentum P and a function $h(\mathbf{P})$ acting as the kinetic energy in classical mechanics. Therefore, $h(\mathbf{P}) = \mathbf{P}^2/(2m)$ is a natural choice which also results in a Gaussian distribution of $exp(-h(\mathbf{P}))$. m is a chosen constant as an analogy to the mass in classical mechanics. The Hamiltonian becomes

$$H(\mathbf{X},\mathbf{P}) = A(\mathbf{X}) + \frac{1}{2m}\mathbf{P}^2 \tag{3.9}$$

Then, starting from an initial condition $\{\mathbf{X}(0),\mathbf{P}(0)\}$, we can find $\{\mathbf{X}(s),\mathbf{P}(s)\}$ by integrating Eq. 3.9. Similar to PAMC, this proposed move is then accepted with a probability

$$P = min(1, exp[-H(\mathbf{X}(s), \mathbf{P}(s)) + H(\mathbf{X}(0), \mathbf{P}(0))]) \tag{3.10}$$

which theoretically should be equal to 1, since by following Eq. 3.9, $H(\mathbf{X}, \mathbf{P})$ is conserved.

In practice, however, we need to discretize s in order to perform the integration. Previous study by Zhong and Marsden [33] has shown that, when we use a symplectic integrator as described in [15], the symplectic form and Hamiltonian cannot be both preserved [33]. As a result, we could expect $H(\mathbf{X}(s), \mathbf{P}(s)) \approx H(\mathbf{X}(0), \mathbf{P}(0))$, but not equal exactly. Then the real acceptance rate will be close to, but not equal to 1.

In the actual implementation of HMC that we have used, among all the symplectic integrator available [22] [15], we chose the commonly used leapfrog integrator, due to its simplicity and accuracy. To integrate from $\{\mathbf{X}(0), \mathbf{P}(0)\}$ to $\{\mathbf{X}(s), \mathbf{P}(s))\}$, we need to choose an integration step size ε and number of steps N_s, so that $\varepsilon N_s = s$. Using the leapfrog symplectic stepping rule, we can move to $\{\mathbf{X}(\varepsilon), \mathbf{P}(\varepsilon)\}$:

$$\begin{cases} \mathbf{P}(\varepsilon/2) = \mathbf{P}(0) - \frac{\varepsilon}{2}\nabla A(\mathbf{X}(0)) \\ \mathbf{X}(\varepsilon) = \mathbf{X}(0) + \varepsilon \mathbf{P}(\varepsilon/2) \\ \mathbf{P}(\varepsilon) = \mathbf{P}(\varepsilon/2) - \frac{\varepsilon}{2}\nabla A(\mathbf{X}(\varepsilon)) \end{cases} \tag{3.11}$$

Subsequently, we can continue moving to the next position $\{\mathbf{X}(2\varepsilon), \mathbf{P}(2\varepsilon)\}$. After N_s steps, we will move to the desired point $\{\mathbf{X}(s), \mathbf{P}(s))\}$.

acknowledgements

Chapter 3, in part, has been adapted from Z. Fang, A. S. Wong, K. Hao, A. J. Ty, and H. D. Abarbanel. Precision annealing monte carlo methods for statistical data assimilation and machine

learning. *Physical Review Research*, 2(1):013050, 2020 and A. S. Wong, K. Hao, Z. Fang, and H. D. Abarbanel. Precision Annealing Monte Carlo Methods for Statistical Data Assimilation: Metropolis-Hastings Procedures. arXiv preprint arXiv:1901.04598 (2019). The author of this thesis was a co-first author of these papers.

Chapter 4

Lorenz 96 Model

4.1 Model Description

Lorenz 96 model is an atmospheric model formulated by Lorenz in 1996 [18]. A D-dimensional Lorenz 96 model satisfies following equations:

$$\frac{dx_a(t)}{dt} = x_{a-1}(t)(x_{a+1}(t) - x_{a-2}(t)) - x_a(t) + F \tag{4.1}$$

where $a = 1, 2, \ldots, D$ for any $D > 3$. This model is a simplified version to simulate some quantity in the atmosphere of a latitude circle. The quadratic term and linear term represent advection and dissipation respectively [18]. The model follows periodic boundary conditions, which means $x_{a+D}(t) = x_{a-D}(t) = x_a(t)$, since it is on a circle. F is the constant forcing parameter of the model.

Lorenz 96 is a very good instructional model to test data assimilation method because of the following properties.

1. While the form of the equations is simple, it simulates the three essential parts of physics in the atmosphere: advection, dissipation and external forcing.

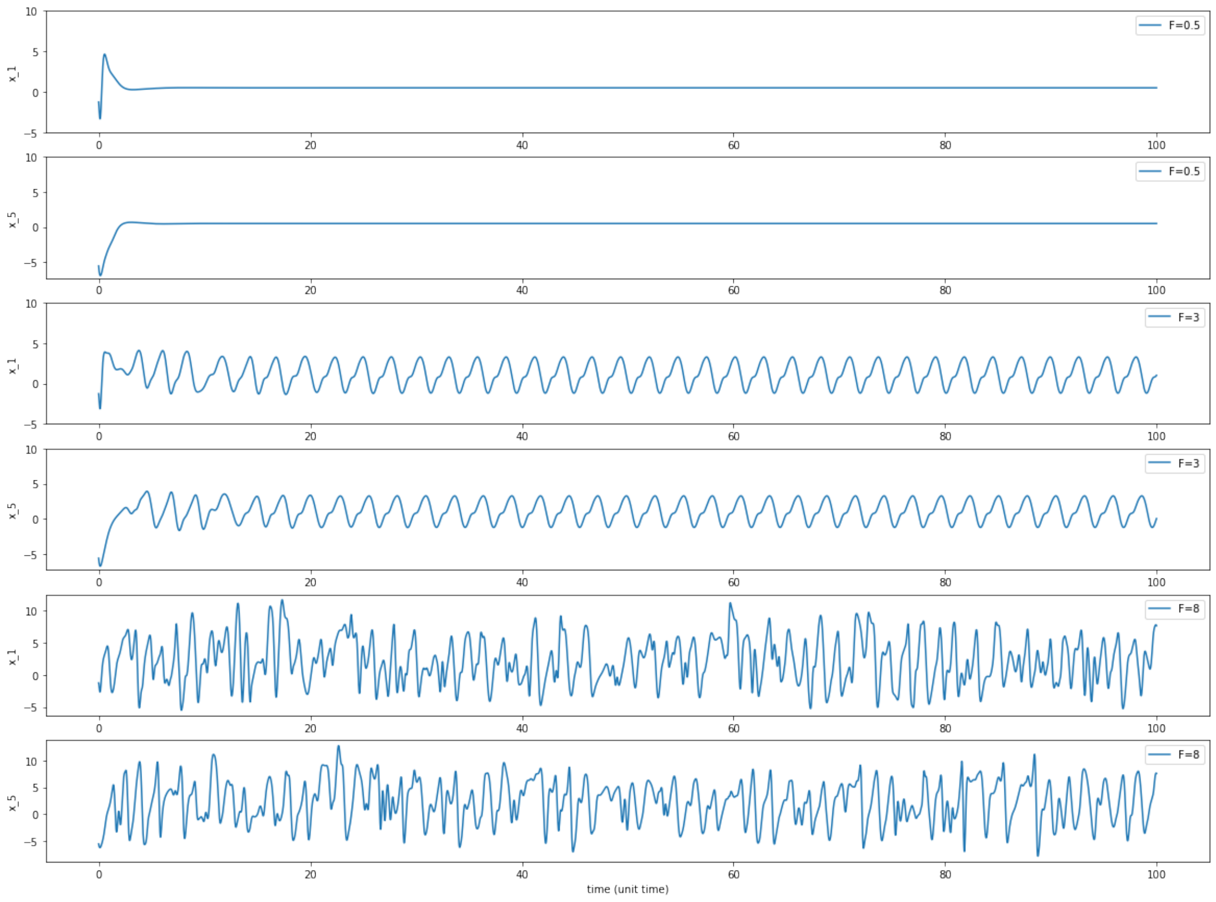


Figure 4.1: Times series of Lorenz 96 model with different forcing parameters. In the first two graphs, $F = 0.5$ and the model decays to the equilibrium positions. In the third and fourth graphs, $F = 3$ and the model shows a periodic behavior. In the last two graphs, $F = 8$ and the model is in chaotic regime.

2. Its dimension D is adjustable, as we can arbitrarily divide the circle into as many different sections as we want. So we can start with a small value to validate the method, then increasing D to easily evaluate how well the method scales as the dimension of the model increases.

3. By tuning the parameter F, the model will exhibit different behavior. When F is very small, all the state variables will decay to the equilibrium position $X_a = F$ for all $a = 0, 1, \ldots, D$. As F becomes large, the model will have periodic solutions. When F is large, roughly $F \geq 8$, the model is chaotic. An illustration of the three different behaviors is shown in Fig. 4.1.

4. All the variables are on the same scale. As a result, there is no need to tune any hyper-parameter for each individual variable.

4.2 Twin Experiments Setup

In this chapter, I will discuss the results of twin experiments on Lorenz 96 model using PAMC. The parameters used for PAMC and the model in the twin experiments are summarized in Table 4.1.

The actual paths for twin experiments is generated by integrating forward using the 4th order Runge-Kutta method. Then the Gaussian noise with 0 mean and 1.0 standard deviation is added to the paths, and these noisy variables are partly taken, based on the number of observed variables L, as the measurements. Then the PAMC is performed using the measurements and the model we have seen in section 4.1, with the listed hyper-parameters in Table 4.1.

Table 4.1: A table to summarize the setup of twin experiments for Lorenz 96 model.

Parameter	Value	Description
D	20	Dimension of the model
L	12, 9, 5	Number of observed dimensions
F	8.17	Forcing parameter of the model
R_m	1.0	$R_m(l)$ in Eq. 2.12. Same for every l
R_{f0}	1.0	R_{f0} as used in Chapter 3. Same for every dimension
δt	0.025	Integration time
M	200	Number of time steps in the measurement window
α	1.2	$R_f = R_{f0}\alpha^\beta$ as described in Chapter 3
β	50	$R_f = R_{f0}\alpha^\beta$ as described in Chapter 3
N_I	50	Number of different initial conditions
$N_{initial}$	1000	Number of iterations during the initialzation phase in each annealing step
N_{it}	5000	Number of iterations in each annealing step
γ	0.3	γ in Eq. 3.2
f_0	0.5	f_0 in Eq. 3.2

4.3 Results Using PAMC

4.3.1 D=20, L=12

When 12 variables out of 20 are observed, Fig. 4.2 shows the action curves as beta increases for 50 different initial conditions. we can find that, no matter where we start within the dynamical range ($[-10, 10]$ for all variables in Lorenz 96 model) as the initial condition, the action approaches the same level asymptotically, which is the theoretical value of the action as $R_f \to \infty$.

In order to show the change in measurement error and model error, the total action, measurement error and model error is plotted in Fig. 4.3 for a randomly chosen initial condition. when beta is small, measurement error and model error are approximately the same order of magnitude, because we start with $R_f = R_m$, and thus the paths found for each annealing step do not follow the model strictly, but stay in a middle ground between matching the measurement

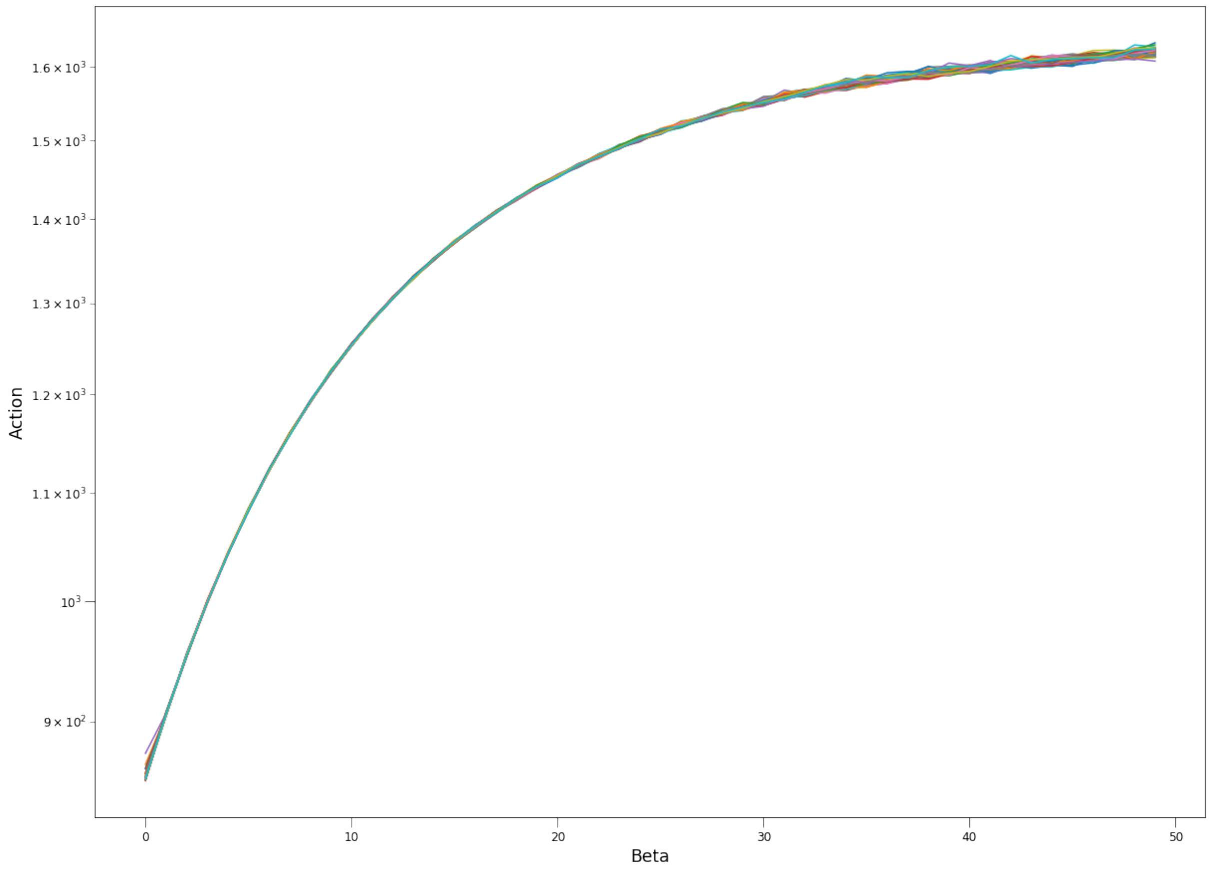


Figure 4.2: Action curves when $D = 20$, $L = 12$. $N_i = 50$ different initial conditions are used. All different initial conditions reach the same asymptotic level as β increases.

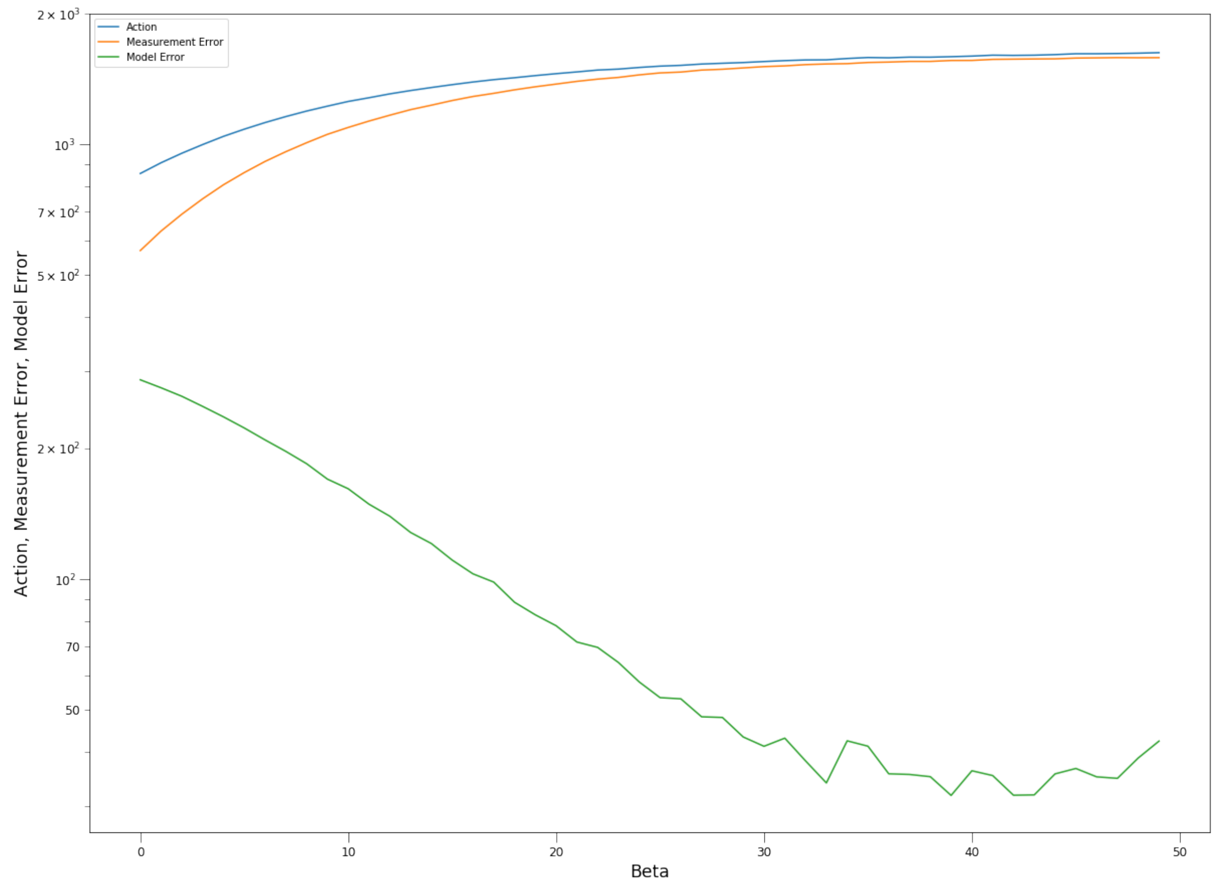


Figure 4.3: Action, measurement error, and model error of one randomly chosen initial condition, $L = 12$, $D = 20$. As β increases, measurement error increases, while model error decreases, until they become independent of beta or R_f.

perfectly and fitting to the model completely.

As beta increases, R_f becomes larger, since $R_f = Rf0\alpha^{\beta}$. The expected paths found after the annealing procedure will then fit to the model more and more strictly and the decrease in model error is therefore observed. Eventually the model error reaches a value that it cannot decrease anymore due to numerical stability. As a result, we will find that both measurement error and model error become independent of R_f, and the model error is about 2-3 order of magnitude smaller even though R_f is much larger than R_m now. This is an evidence showing that we are sampling from the dominating region in calculating the expected value.

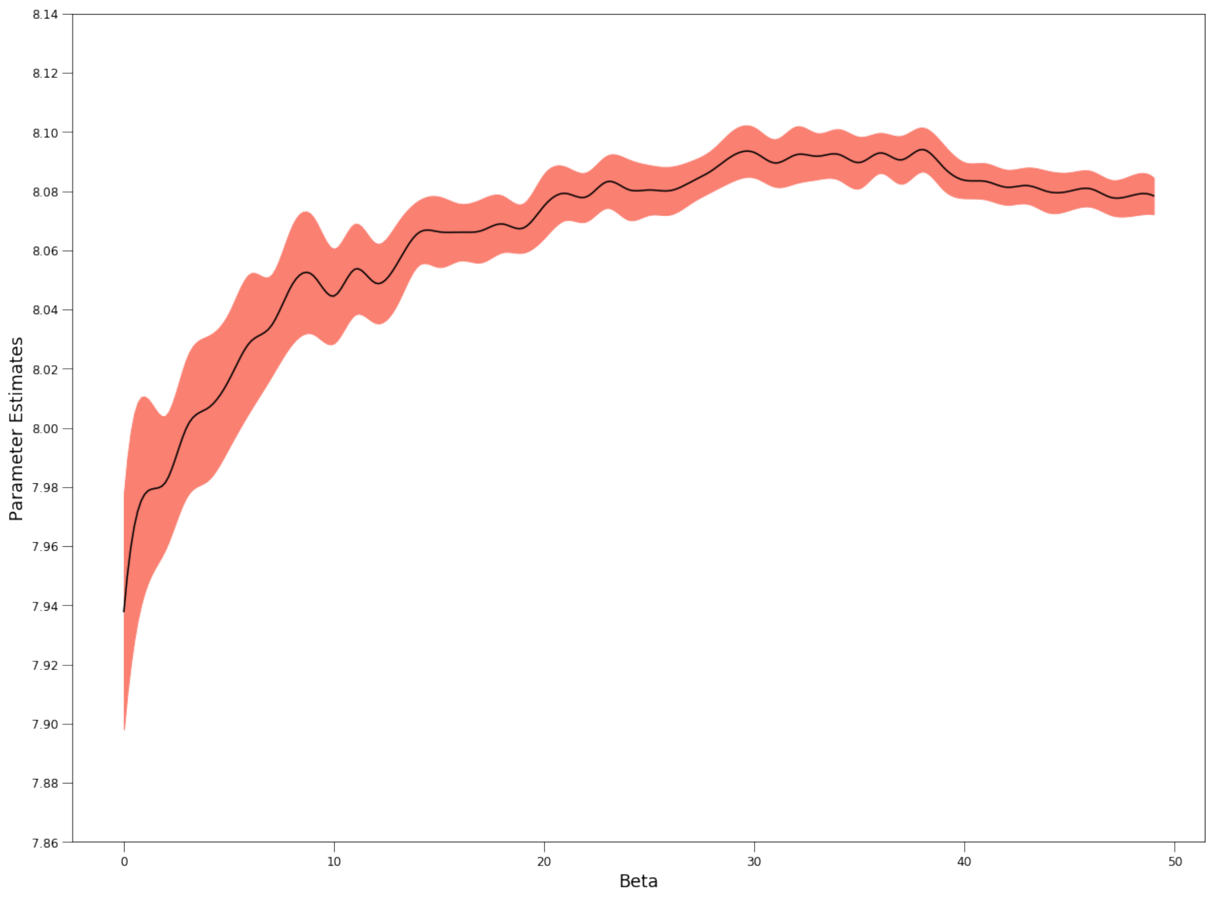


Figure 4.4: Estimation of the forcing parameter, $L = 12$, $D = 20$. Light red area is the error in the estimation, which is the standard deviation of the estimates from $N_i = 50$ different initial conditions. As beta increases, the parameter estimates get closer to the actual value, and the error in the estimates, which is the standard deviation calculated from 50 different initial conditions, becomes smaller.

Fig. 4.4 shows the estimation of the forcing parameter F for all β. The error, which is the light red area, is calculated as the standard deviation from different initial conditions. At the beginning, because we initialize the values of the parameter randomly , the error is large, and the expected value is off from the actual value. The annealing procedure will not help to bring the expected value closer to the real value, since R_f is small, and we are not forcing the estimations strictly to the model. Then as beta increases and the constraint from the model gets larger, similar to the behavior of the actions, our estimation of the parameter becomes closer to the actual value, and the error is getting smaller as well.

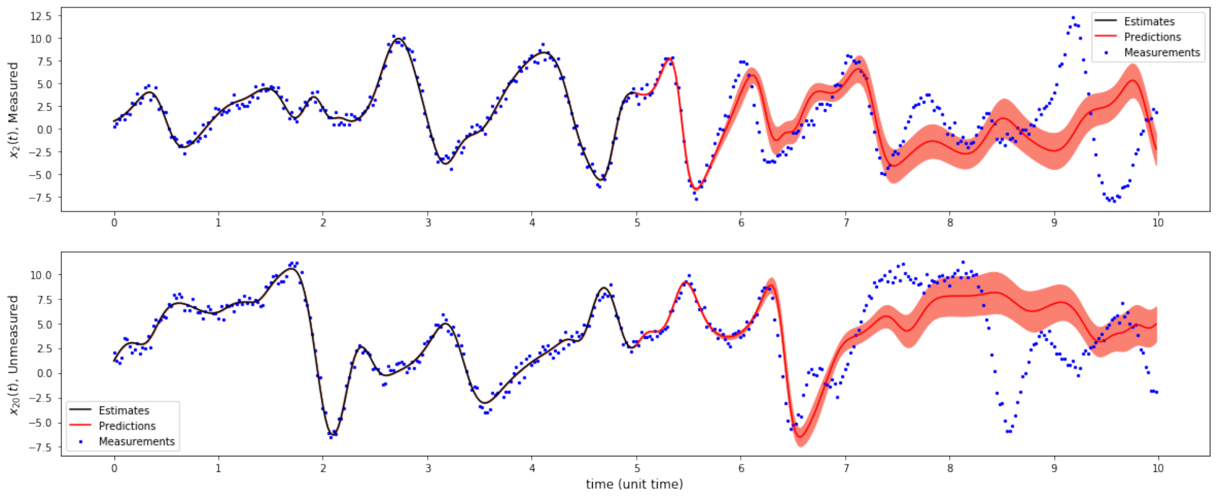


Figure 4.5: Measurements, estimates and predictions of an observed and an unobserved variables, $L = 12$, $D = 20$. Black lines in first 5 unit time are the estimations using weighted averages of paths we get from different initial conditions. Red lines are the predictions given the estimations of variables and forcing parameters. Light red areas represent the error in variables, which are the standard deviations from the paths. blue dots are the measurements. The measurements are plotted for all variables at all times as a comparison to the estimations and predictions, but they are not known in the PAMC procedures for unobserved variables and when making predictions.

After the PAMC procedure is done, we have an estimated path for each initial conditions. Using $exp(-A)$ as weights, we can then find the weighted average as the expected value for any variable. In this case, since the actions for all initial conditions converge to the same level, it indicates that we end up with paths that are very close to each other. This can also be verified in Fig. 4.5.

Fig. 4.5 contains graphs for 2 of the 20 variables, one observed and one unobserved. The error on the graph is the standard deviation calculated from all the estimated paths. During the measurement window, which is the first 5 unit time, the error is very small. This further proves that all the paths converges to the same region that minimizes the action. Also we achieved accurate estimations with small error in both observed and unobserved variables, meaning that 12 measured variables are enough to transfer information from measurements to the model and estimate all the variables and parameters in a 20-dimension Lorenz 96 model.

In the prediction window, we are able to predict fairly close to the actual observations within about 1.5 unit time. Inevitability, the error keeps growing exponentially and predictions fail because of the chaotic nature given the model and forcing parameter we used.

4.3.2 D=20, L=9

When $L = 9$, Fig. 4.6 shows the actions for all 50 initial conditions. In this situation, instead of converging to the same action level, we can find that the actions from different initial conditions split into several different levels. Some of them are sill able to reach the region where action is the lowest. For those curves, the behavior of measurement error and model error are still similar to Fig. 4.3. For other curves, it is clear that the actions are increasing exponentially as beta increases, which suggests that they are trapped in a local minimum where the model error is not 0. The slope of the action curve near the end is proportional to the value of remaining model error.

The estimation of the forcing parameter is plotted in Fig. 4.7. The error here is much greater than the case we have seen previously where $L = 12$, because many of the paths we get do not reach the region with the minimum of action and as a result, the parameter estimates are far off from the actual value for those paths.

Fig. 4.8 shows plots for an observed and an unobserved variables. Compared to 4.5, when we only have 9 measurements, even during the measurement window, the uncertainties in the estimations are large sometimes. During predictions, the average value is still close to the actual value for about 0.5 to 1 unit time, because the dominant contribution to the average values is from those paths that have lowest actions. The error, however, grows much more rapidly than the case where $L = 12$.

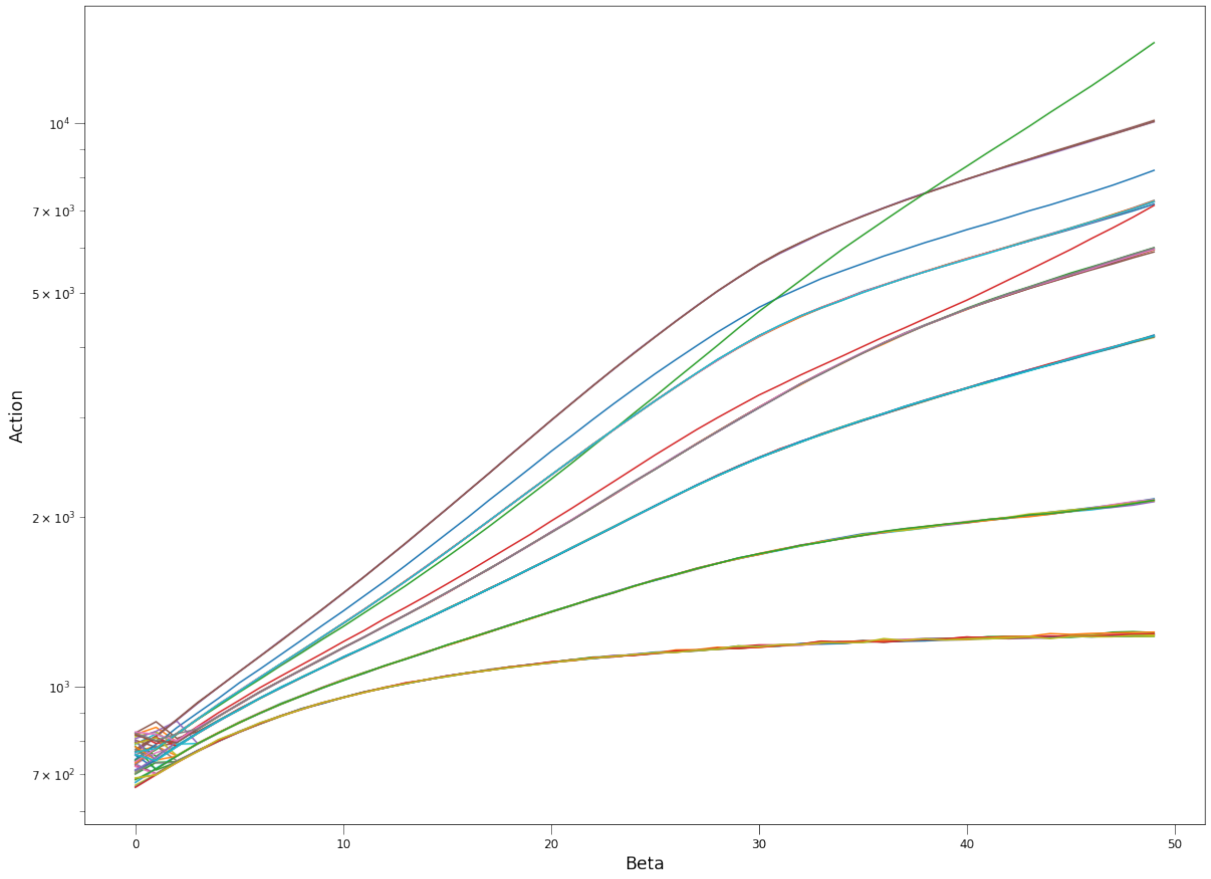


Figure 4.6: Action curves when $D = 20$, $L = 9$. Actions from different initial conditions split into different levels. $N_i = 50$ different initial conditions are used. Some initial conditions reach the level with lowest action value as beta increases, while others are in levels with significantly higher action values.

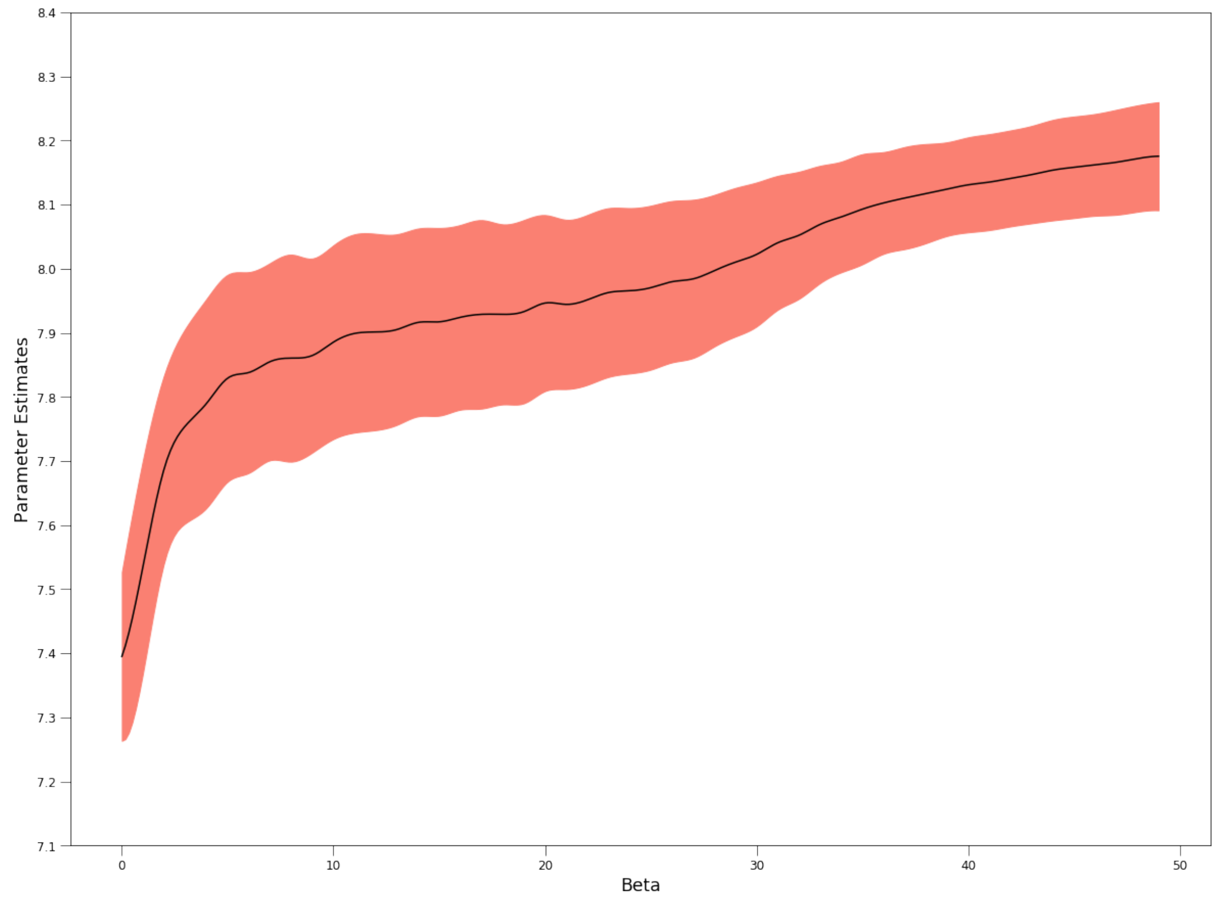


Figure 4.7: Estimation of the forcing parameter, $L = 9$, $D = 20$. Light red area is the error in the estimation, which is the standard deviation of the estimates from $N_i = 50$ different initial conditions. The parameter estimates at large beta is still close to the actual value. However, the error in the estimates is much larger than the case where $L = 12$.

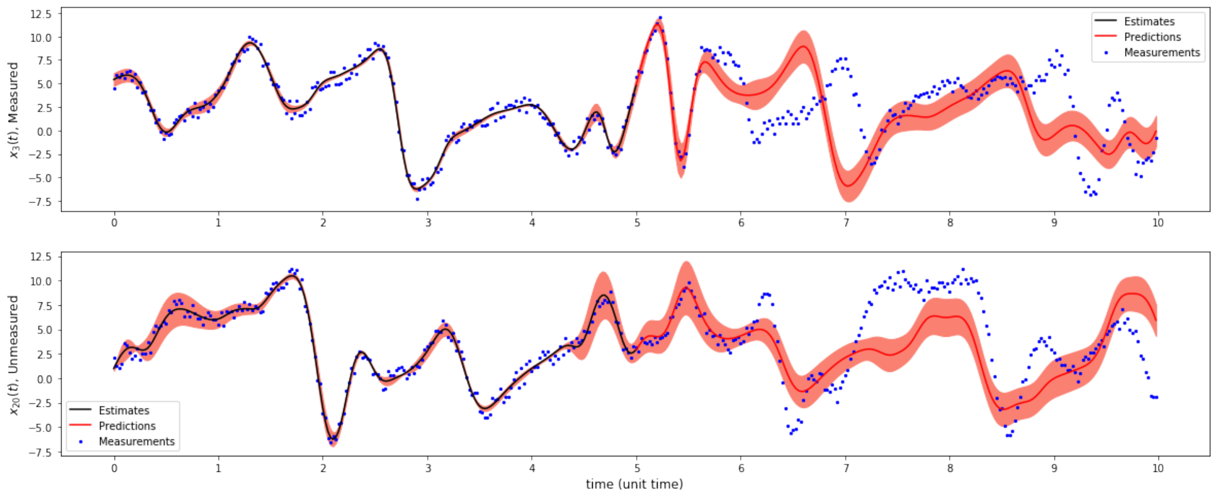


Figure 4.8: Measurements, estimates and predictions of an observed and an unobserved variables, $L = 9$, $D = 20$. Black lines in first 5 unit time are the estimations using weighted averages of paths we get from different initial conditions. Red lines are the predictions given the estimations of variables and forcing parameters. Light red areas represent the error in variables, which are the standard deviations from the paths. blue dots are the measurements. The measurements are plotted for all variables at all times as a comparison to the estimations and predictions, but they are not known in the PAMC procedures for unobserved variables and when making predictions.

4.3.3 D=20, L=5

Finally, we encountered the problem where very few measurements are taken, $L = 5$ in this case. Fig. 4.9 is the action plot for 50 different initial conditions. Since all the curves increase exponentially as beta increases, we can conclude that none of the 50 trials find and sample around the global minimum of the action.

This conclusion can be further supported by Fig. 4.10, where the action, measurement error, and model error are plotted on the same graph for a single initial condition. We can find from the graph that the measurement error and the model error are about the same level. The exponential increase of the model error confirms that the PAMC procedure is sampling around a local minimum, instead of the global minimum as in Fig. 4.3, where the model error become independent of beta at large beta value.

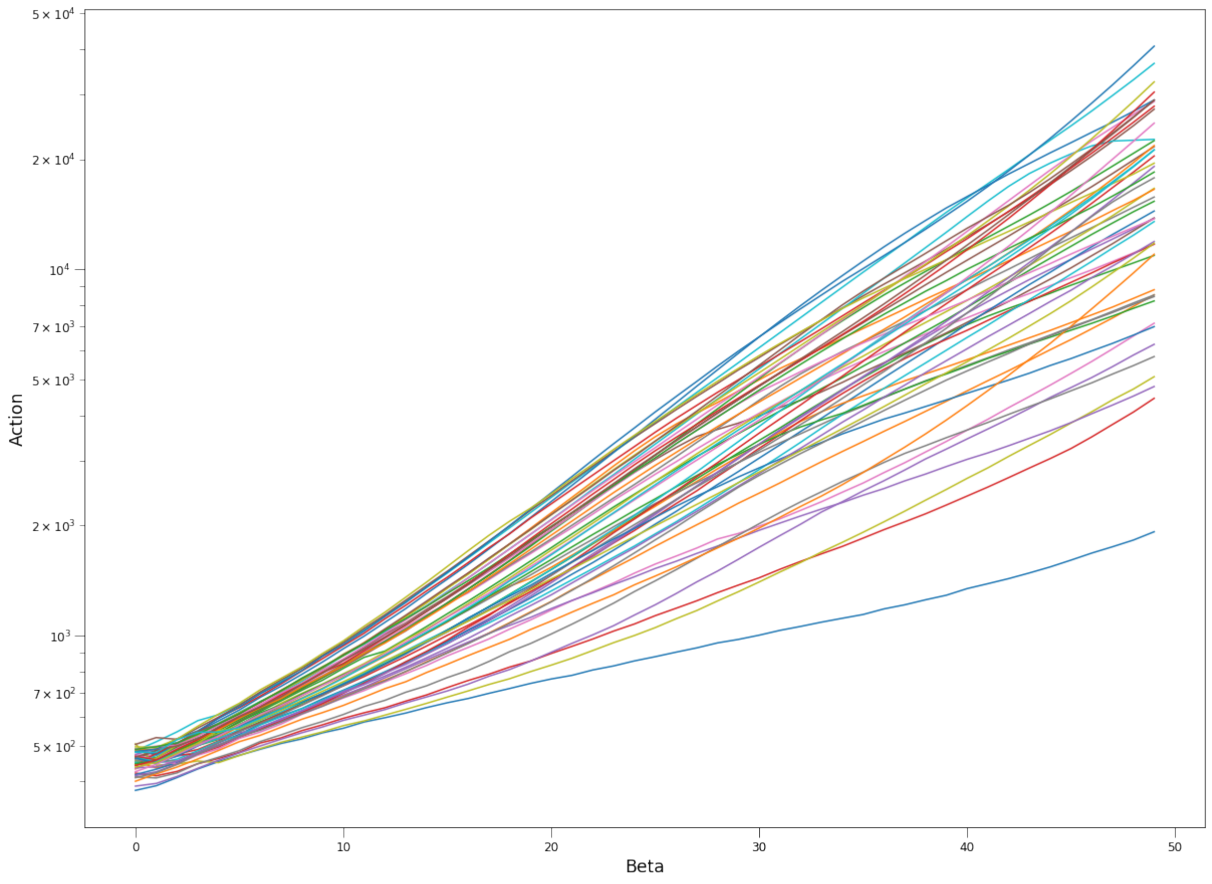


Figure 4.9: Action curves when $L = 5$, $D = 20$. Actions from different initial conditions split into different levels. $N_i = 50$ different initial conditions are used. The actions for all initial conditions grows exponentially as beta increases.

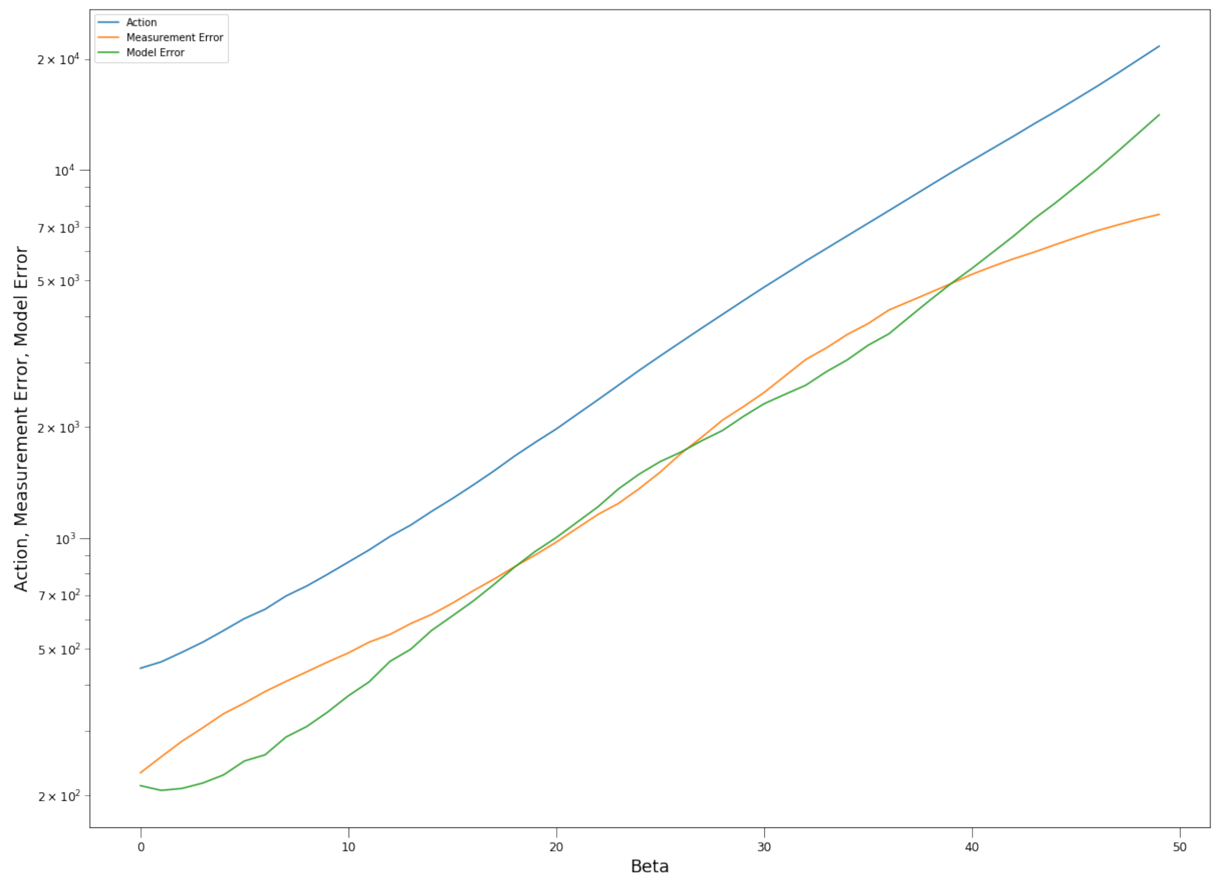


Figure 4.10: Action, measurement error, and model error of one randomly chosen initial condition, $L = 5$, $D = 20$. As beta increases, both the measurement error and the model error increases exponentially.

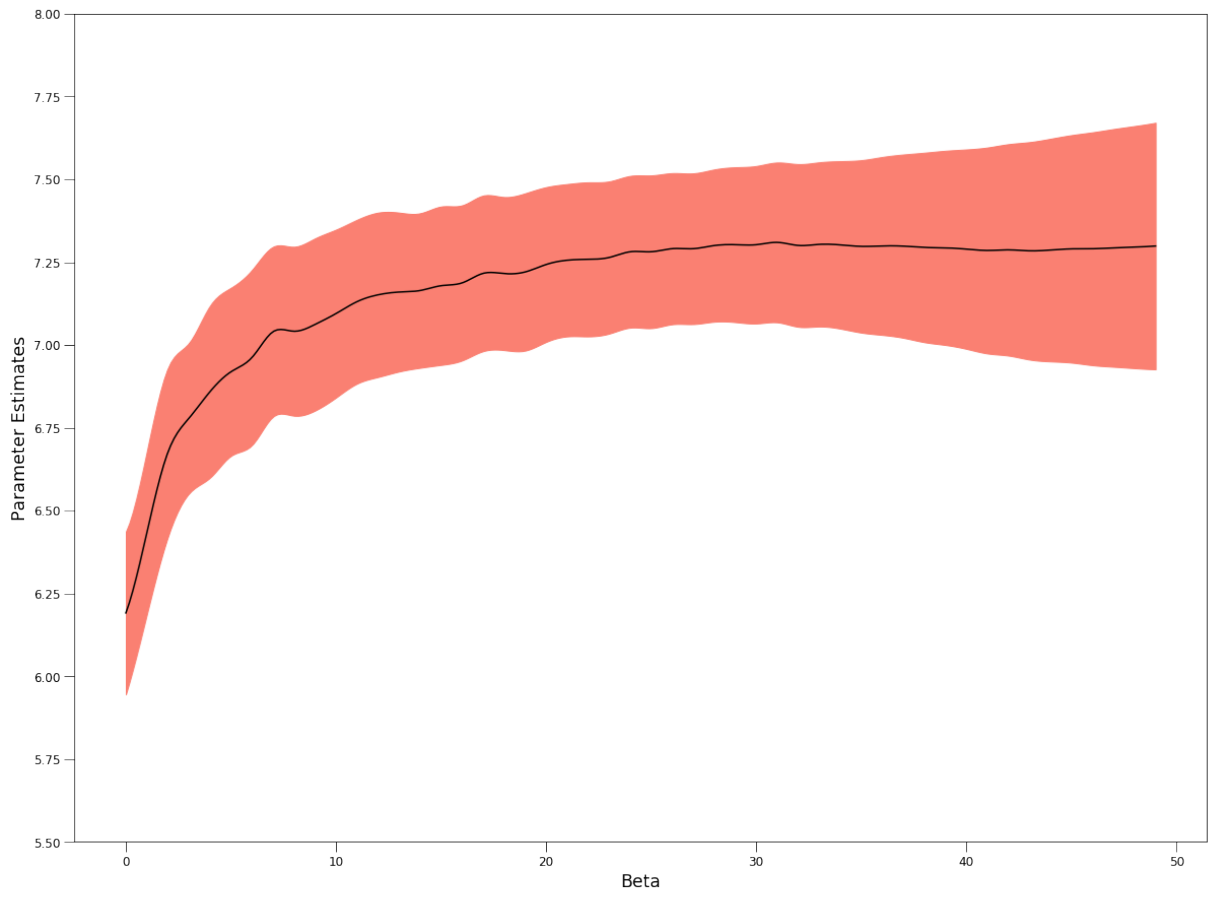


Figure 4.11: Estimation of the forcing parameter, $L = 5$, $D = 20$. Light red area is the error in the estimation, which is the standard deviation of the estimates from $N_i = 50$ different initial conditions. The parameter estimates are not accurate, the error in the estimates is large.

Considering that all the trials are sampling around local minimums, it is reasonable that estimations of the parameter and state variables are not accurate. As shown in Fig. 4.11, the parameter estimates have even larger errors than the case where $L = 9$. and the actual value of the forcing parameter does not fall within the range of uncertainty even at large beta.

Now we can pay our attention to the estimations of state variables, which are plotted in Fig. 4.12. The error is consistently large for $L = 5$, as most of the initial conditions end up in the region where the model error is large. As expected, both estimates and prediction are off from the actual value most of the time.

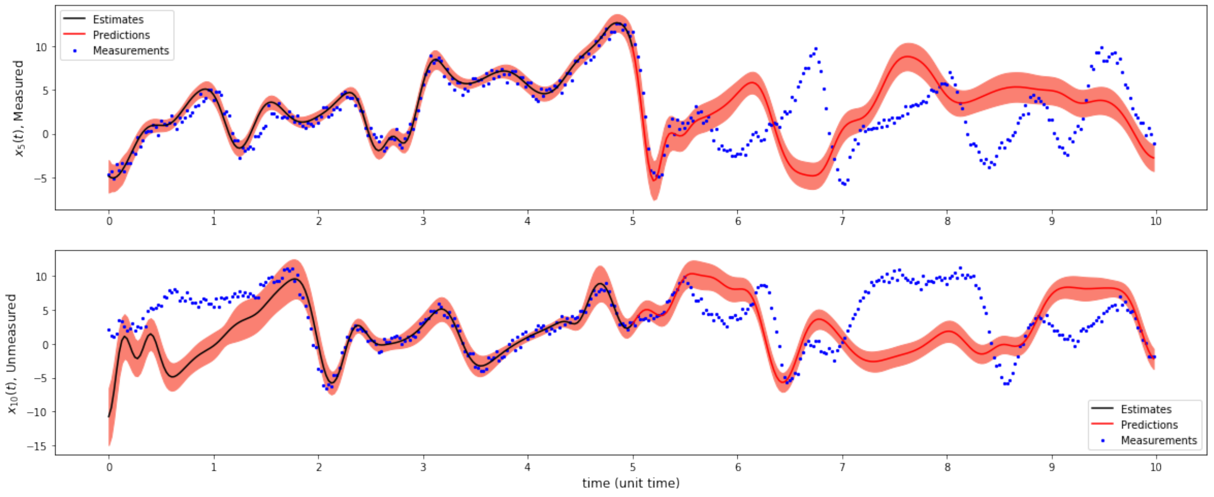


Figure 4.12: Measurements, estimates and predictions of an observed and an unobserved variables, $L = 5$, $D = 20$. Black lines in first 5 unit time are the estimations using weighted averages of paths we get from different initial conditions. Red lines are the predictions given the estimations of variables and forcing parameters. Light red areas represent the error in variables, which are the standard deviations from the paths. blue dots are the measurements. The measurements are plotted for all variables at all times as a comparison to the estimations and predictions, but they are not known in the PAMC procedures for unobserved variables and when making predictions.

All the above plots for actions, estimated parameters, and state variable estimations and predictions are strong pieces of evidence that 5 observed variables are not enough for successful estimations and predictions.

4.3.4 Summary of the Results

In this section, The results of applying PAMC method on a 20-dimension Lorenz 96 model are shown. We have examined the cases for different number of measurements: $L = 12, 9, 5$. When $L = 12$ where a adequate number of variables are observed, it can be shown that no matter where we start as the initial condition within the dynamical range, the PAMC procedure will sample around the global minimum. Thus the model error term in the action will become 0, and the action will be independent from the value of beta. Consequently, good estimates of the forcing parameter and state variables can be found.

When the number of observed variables decrease to 9 or 5, at the end of the PAMC procedure, more and more likely it will be trapped in a local minimum instead of sampling around the global minimum. A strong indicator is the exponential increase of the action and the model error as beta increases. As a result, the estimates will be off from the actual values, and the uncertainty in the estimates will be larger as well. Then we can conclude that we do not have enough measurements to successfully transfer the information from the measurements to the model, and make accurate estimations and predictions in these situations.

acknowledgements

Chapter 4, in part, has been adapted from Z. Fang, A. S. Wong, K. Hao, A. J. Ty, and H. D. Abarbanel. Precision annealing monte carlo methods for statistical data assimilation and machine learning. *Physical Review Research*, 2(1):013050, 2020 and A. S. Wong, K. Hao, Z. Fang, and H. D. Abarbanel. Precision Annealing Monte Carlo Methods for Statistical Data Assimilation: Metropolis-Hastings Procedures. arXiv preprint arXiv:1901.04598 (2019). The author of this thesis was a co-first author of these papers.

Chapter 5

Other Models with More Complicated Features

5.1 Shallow Water Model

5.1.1 Model Description

Shallow water equations are approximations of Navier-Stokes equations when the horizontal scale of the fluid is much larger than the vertical scale. Since the height of the ocean (1-10 km) is much smaller than the earth's radius (about 6000 km), this is a good approximation when studying a large area of ocean, for example, the current in the whole Pacific ocean.

Following [30], the Coriolis force, bottom friction and surface wind stress are included in the model. Therefore, the shallow water equations shall be written as:

$$\frac{\partial h(\vec{r},t)}{\partial t} + \nabla[h(\vec{r},t)\vec{u}(\vec{r},t)] = -\hat{z}\cdot\nabla\times[\frac{\tau(\vec{r})}{\rho f(\vec{r})}] \tag{5.1}$$

$$\frac{\partial\vec{u}(\vec{r},t)}{\partial t}+\vec{u}(\vec{r},t)\cdot\nabla\vec{u}(\vec{r},t)=-g\nabla h(\vec{r},t)+\nu\nabla^2\vec{u}(\vec{r},t)-\varepsilon\vec{u}(\vec{r},t)+\vec{u}(\vec{r},t)\times[\hat{z}f(\vec{r})] \tag{5.2}$$

Here $\vec{r}$ is the horizontal coordinate (x,y), $h(\vec{r},t)$ is the height, and $\vec{u}(\vec{r},t)$ is the 2-D horizontal velocity. $f(\vec{r})$ is the Coriolis parameter and is approximated as $f(\vec{r})=f_0+\beta y$. ε is the Rayleigh friction coefficient related to the friction at the bottom. ν is the kinematic viscosity. τ is the surface stress.

The actual integration and discretization of the grid points follows the procedure in [26], which uses staggered grid to reduce two-grid interval noise, and involves the calculation of mass fluxes and a quantity related to local pressure. The details of this integration scheme will be discussed in the Appendix.

5.1.2 Twin Experiment Setup

To prepare for the twin experiment, we first starting from a random initial condition that obeys the physics laws, then integrated for 500 hours as the spin up period. Then the model is integrated for another 10 hours as the actual solution of the system. Afterwards, the Gaussian noise, which is about 2% of the fluctuation of variables, is added to the actual solution, to be used as our measurements during the time window. The initial conditions of the measurement window are shown in Fig. 5.1.

The model is integrated using second order Runge-Kutta method, with a time step of 9 second. This choice of integration time step is almost the largest we found which will guarantee that the integration is numerically stable. However, this choice in deed limits the length of the

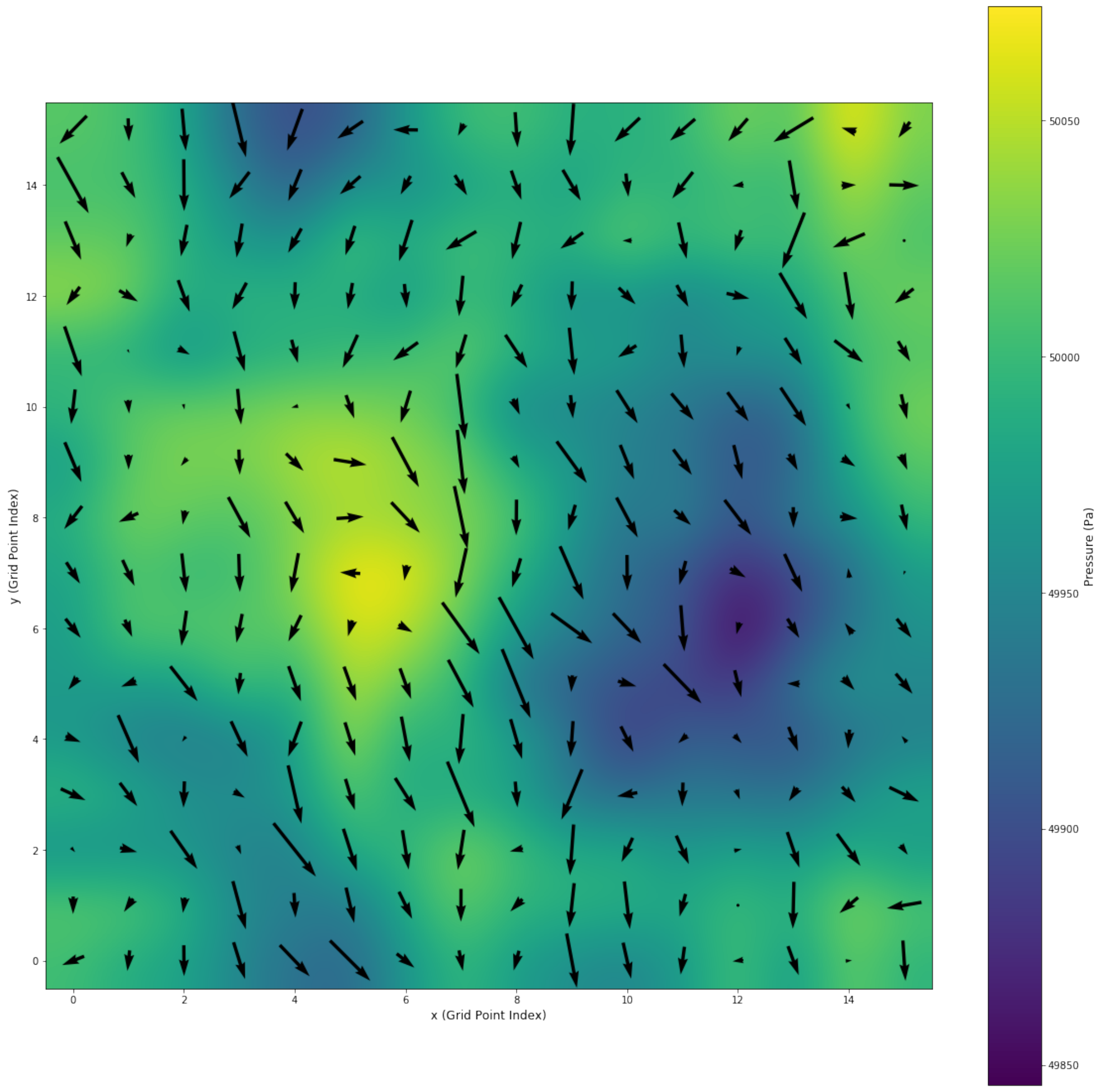


Figure 5.1: Initial conditions of shallow water model in the twin experiment setup. Arrows represent the initial velocities at each grid points, and the color bar shows the pressure, which is proportional to the surface height via equation $p = \rho g h$.

measurement window due to the availability of computing resource. We will talk about this in more detail later in the chapter.

As for the parameters of the model, τ is taken in form $\tau = (\tau_0 cos(2\pi y/L_y), 0)$, where L_y is the total length in north-south direction. Table 5.1 summarizes the meaning of all the parameters in Eq. 5.1 and 5.2 as well as other relevant parameters, including their values used in our simulation.

Table 5.1: A table to list the parameters of shallow water model.

Parameter	Value	Description
D	16	Dimension of the system in each horizontal direction
dt	9s	Integration time step
dx/dy	50km	East-West/North-South grid spacing
h_0	5.1km	Equilibrium Depth
f_0	$5 \times 10^{-5} s^{-1}$	f_0 in Coriolis parameter f
β	$2 \times 10^{-11} (s \cdot m)^{-1}$	β in Coriolis parameter f
ν	$1 \times 10^{-4} m^2 s^{-1}$	Kinematic viscocity
ε	$2 \times 10^{-8} s^{-1}$	Rayleigh friction coefficient
τ_0	$0.002 m^2 s^{-3}$	Amplitude of the surface stress

The 10-hour measurement window is used for the data assimilation procedure. The measurements are observed every 9 second, which is the same as the integration time step. This choice of time interval between measurements is necessary, because in the model error term in Eq. 2.12 requires integrating from a time step to the next time step of measurements. Then if the time interval between measurements is longer, due to the numerical issue, the model error term will not go to zero in the annealing method, and the action will increase exponentially as we increase the R_f value, which may lead to bad estimations and predictions.

5.1.3 Results using Precision Annealing HMC Method

For the shallow water model, the precision annealing HMC method is used instead of the PAMC method, due to the high-dimensional nature of the model. As we have mentioned in Chapter 3, when the dimension gets higher, the acceptance rate will become lower in the PAMC method. In this case, we have tested the PAMC method on the model, and concluded that it is not realistic to finish the procedure in a reasonable time.

The HMC method introduces several extra hyperparameters, which in this case, needs to be tuned for each annealing step. Table 5.2 lists the values or the ranges of all the hyperparameters that are used.

Table 5.2: A table for the hyperparameters of shallow water model, using precision annealing HMC method.

Hyperparameters	Values	Description
R_m	100, 0.25	$R_m(l)$ in Eq. 2.12. 100 for velocities and 0.25 for pressure.
R_{f0}	50, 0.125	R_{f0} as used in Chapter 3. 50 for velocities and 0.125 for pressure.
M	4000	Number of time steps in the measurement window
α	2	$R_f = R_{f0}\alpha^\beta$ as described in Chapter 3
β	18	$R_f = R_{f0}\alpha^\beta$ as described in Chapter 3
N_s	$200 - 500$	Number of integration steps for each proposal
ε	$7.0 \times 10^{-4} - 1.0 \times 10^{-2}$	integration step size
N_{iter}	$200 - 500$	Number of proposals in an annealing step

Unlike our results in Chapter 4, where 50 different initial conditions were used, we only did experiment with a single initial condition for the shallow water model, due to the limit of computing time and resource. However, the results is similar to those in Chapter 4.

When we observed 75% of the total variables, the action plot is shown in 5.2. Note that in

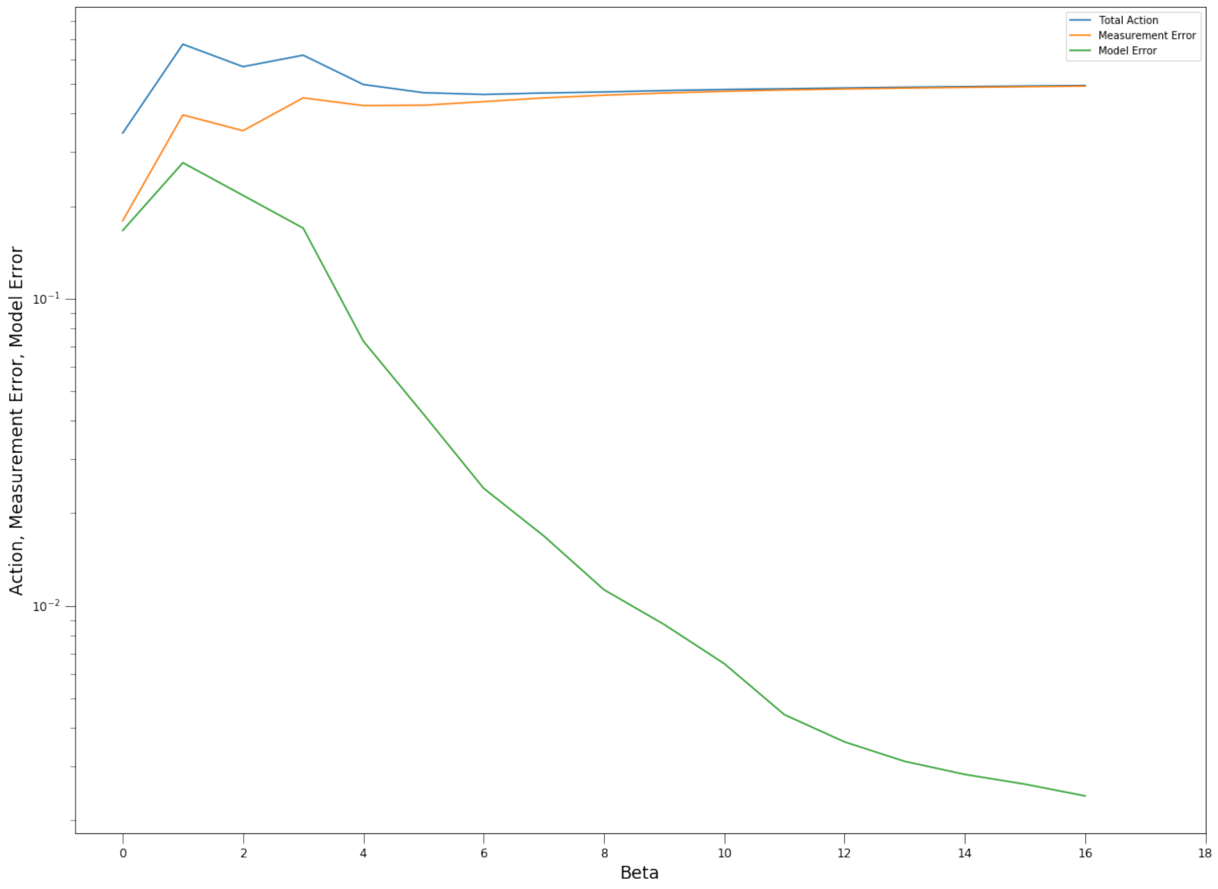


Figure 5.2: Action, measurement error, and model error of the shallow water model, $L = 588$, $D = 768$. As β increases, measurement error increases as well, while the model error decreases, until about 2 order of magnitude smaller than the measurement error in the end.

the setup of the precision annealing HMC method, the measurement error and model error are normalized so that the expected value is independent from the number of dimensions.

In the case of approximately 75% of the variables observed, the measurement error increases as the R_f increases with larger β, and asymptotically reaches the expected value 0.5. Meanwhile, the model error keeps decreasing, becoming 2 order of magnitude smaller than the measurement error at the end of the annealing procedure, which means that there are enough observations to successfully estimate the state variables.

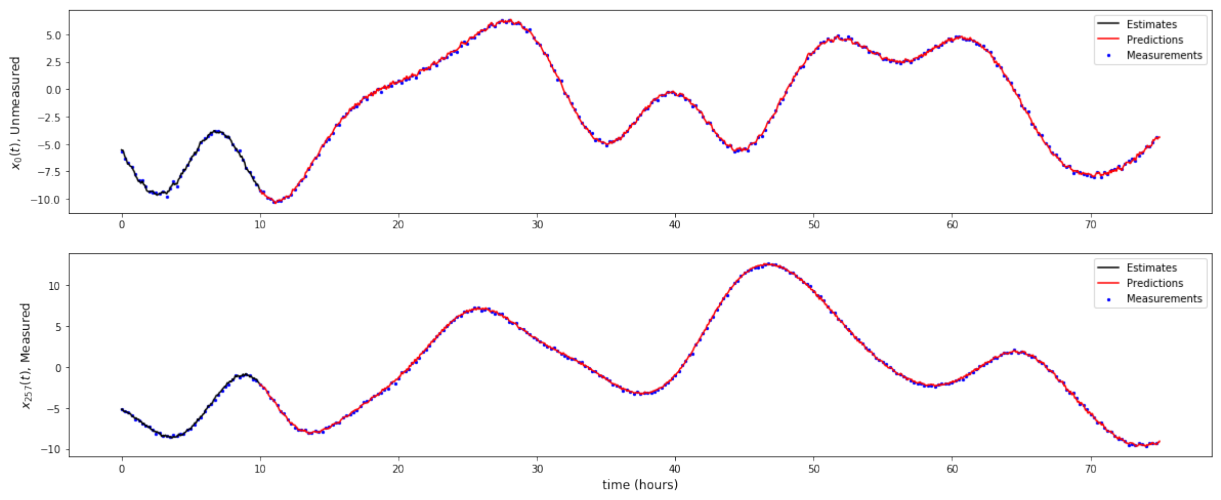


Figure 5.3: Measurements, estimates and predictions of an observed and an unobserved variables, $L = 588$, $D = 768$. Black lines are the estimations during the measurement window. Red lines are the prediction after the measurement window. Blue dots are the measurement for the entire period.

Fig. 5.3 shows the estimated and predicted paths for two state variables, one measured, one unmeasured. Since we have enough measurements, as indicated in Fig. 5.2, The estimations and predictions are accurate for both variables. However, because no standard deviation of the estimations are calculated during the HMC procedure for memory efficiency and we only use one initial condition, we are therefore not able to find the error in the estimations and predictions and show the accuracy of the estimated and predicted paths.

When the number of measurements deceases to about 70% of all the variables, the graph for actions is presented in Fig. 5.4. We can see that the model error increases in the middle of our annealing steps, which means that it is likely to be trapped in a local minimum during those steps. However, since HMC can make large moves in the variable space between proposals, eventually it escapes from the local minimum and reaches to the point where model error is much smaller than the measurement error. Then the estimations and predictions are still accurate at the end of our annealing procedure.

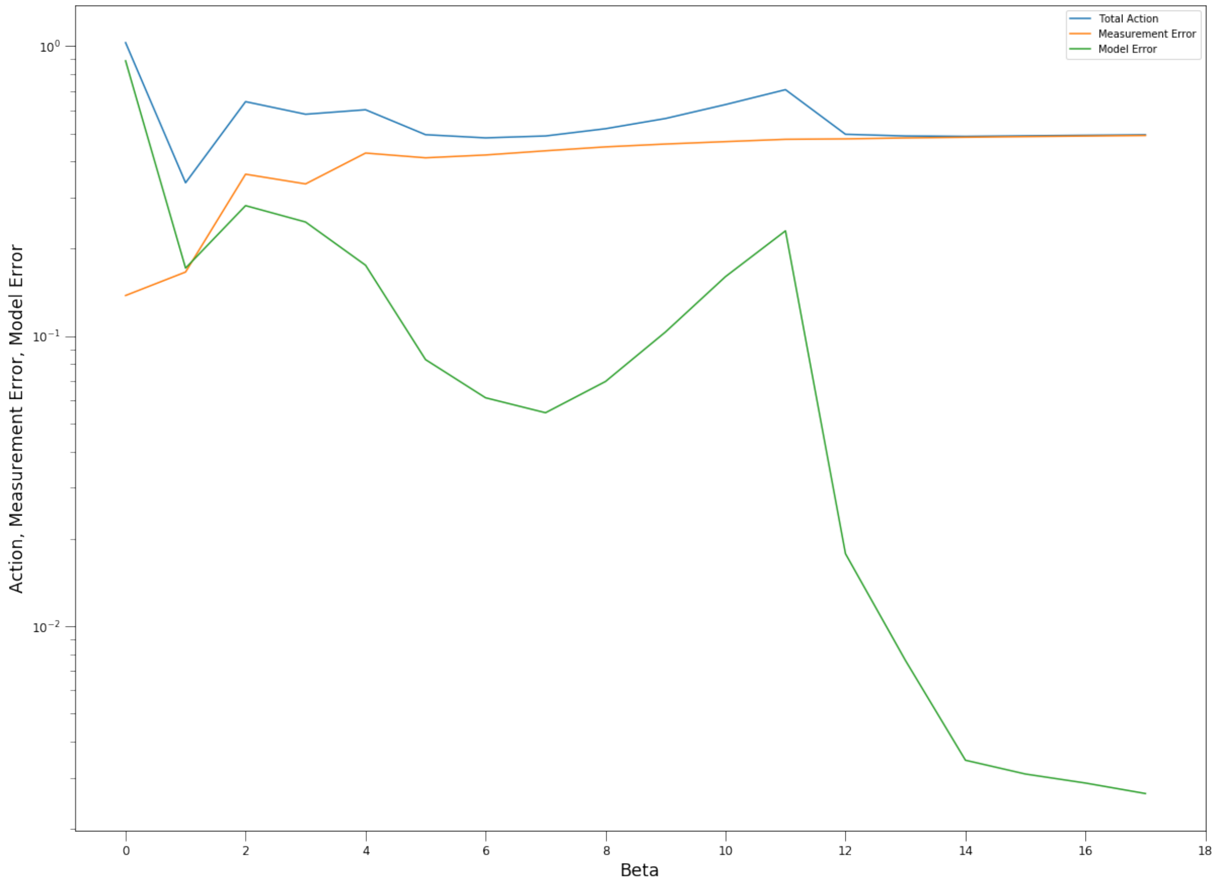


Figure 5.4: Action, measurement error, and model error of the shallow water model, $L = 550$, $D = 768$. As β increases, measurement error increases, but the model error increases in the middle of the procedure, but eventually reaches a low level.

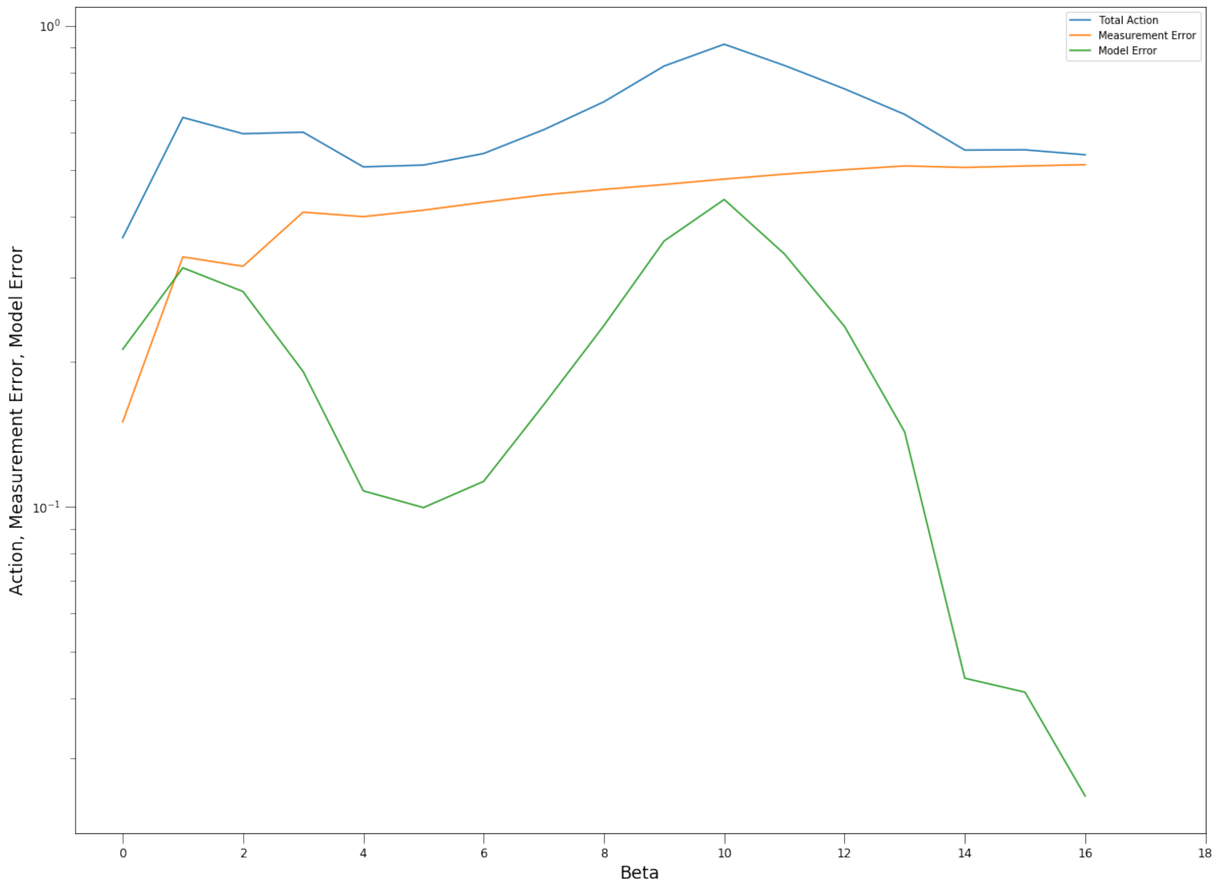


Figure 5.5: Action, measurement error, and model error of the shallow water model, $L = 512$, $D = 768$. As β increases, measurement error increases, but the model error increases in the middle of the procedure, but eventually reaches a low level.

For the last case, 66% of the variables are measured. The action, measurement error and model error are plotted in Fig. 5.5. The behavior is similar to the situation where we have 70% measurements, but at the end the model error is not as small.

Fig. 5.6 shows the paths of an measured and an unmeasured variables with 66% observations. As we can see, even though the model error is still small compared to the measurement error, because it does not reach the same level as when we have 70% and 75% of measurements, the predictions diverges from the observations in about 5-10 hours after the measurement window. It indicates as well that our annealing procedure has not found good enough estimations of the variables at the end.

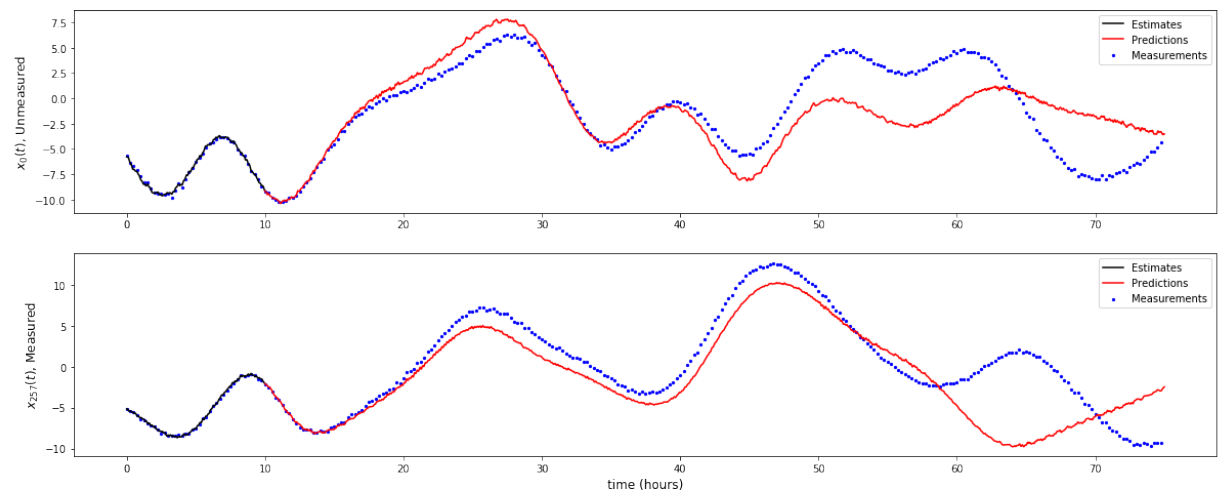


Figure 5.6: Measurements, estimates and predictions of an observed and an unobserved variables, $L = 588$, $D = 768$. Black lines are the estimations during the measurement window. Red lines are the prediction after the measurement window. Blue dots are the measurement for the entire period.

5.1.4 Discussion on the Computing Issue of High-Dimensional Models

In the shallow water model, the dimension of the model is 768, and the number of time steps we used is 4000. Compared to the Lorenz 96 model we have seen in Chapter 4, where the numbers are 20 and 200 respectively, the total number of variables is roughly 800 times larger. This difference is one of the main motivation for us to choose the HMC method. For the implementation of the HMC method, the PyTorch library in Python is used for two reasons. First it supports the use of GPU to speed up the calculation, and second it has an Autograd module that can calculate the gradient given the function, which is a convenient tool as the gradient is needed in the HMC method.

However, with the current implementation of the HMC method, the computing time is still long. In general, the time it takes to finish a single annealing step varies between 3-10 hours, depending on the specific hyperparameters of the setup. Therefore, only a single trial of initial condition was performed in this section, as a demonstration to show the effectiveness of the method on a more complicated model. If more GPUs are available, similar to the experiment in Chapter 4, a study with many different initial conditions can be done, and we may have a better idea on how the action plots look like and when the split among action levels of different trials happens.

Moreover, the limited memory of GPU is also a problem. A GeForce GTX 1080 graphic card is used for the calculation, which has 8 GB of memory. With this amount of memory available in GPU, we can only do calculations on the shallow water model with a time window of 4000 steps. Otherwise the GPU will run out of memory and raise an error after exiting the code.

5.2 Regional Ocean Modeling System

5.2.1 Model Description

Regional Ocean Modeling System (ROMS) is an ocean model that has a wide range of applications. Its governing dynamical equations are three-dimensional, free-surface, Reynolds-averaged Navier-Stokes equations [9]:

$$\frac{\partial u}{\partial t} + \vec{v} \cdot \nabla u - f v = -\frac{\partial \phi}{\partial x} - \frac{\partial}{\partial z}\left(\overline{u'w'} - \nu \frac{\partial u}{\partial z}\right) + \mathcal{F}_u + \mathcal{D}_u \tag{5.3}$$

$$\frac{\partial v}{\partial t} + \vec{v} \cdot \nabla v + f u = -\frac{\partial \phi}{\partial y} - \frac{\partial}{\partial z}\left(\overline{v'w'} - \nu \frac{\partial v}{\partial z}\right) + \mathcal{F}_v + \mathcal{D}_v \tag{5.4}$$

$$\frac{\partial \phi}{\partial z} = -\frac{\rho g}{\rho_o} \tag{5.5}$$

as well as the transports equation for tracers like temperature, salinity:

$$\frac{\partial C}{\partial t} + \vec{v} \cdot \nabla C = -\frac{\partial}{\partial z}\left(\overline{C'w'} - \nu_\theta \frac{\partial C}{\partial z}\right) + \mathcal{F}_C + \mathcal{D}_C \tag{5.6}$$

The model also satisfies the continuity equation:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{5.7}$$

The variables and parameters appeared in Eq. 5.3 - Eq. 5.7 are listed in Table 5.3

Table 5.3: A table to summarize the variables and parameters in the equations of ROMS model.

Variables/Parameters	Description
u, v, w	Velocity components in x, y, z directions
C	The tracer of interest
u', v', w', C'	Turbulent fluctuations of u, v, w, and C
ϕ	Dynamic pressure, $\phi = P/\rho_0$, where P is the pressure
f	Coriolis parameter
F_u, F_v, F_C	Forcing or source terms for u, v, and C
D_u, D_v, D_C	Diffusion terms for u, v, and C
$\rho_0 + \rho$	Total *in situ* density
ν	Molecular viscosity
ν_θ	Molecular diffusivity

In these equations we have also made a few assumptions and approximations:

- Time-averaged equations. The idea is based on Reynolds decomposition, where a variable is decomposed into a time-averaged term and a fluctuating term [25]. For example, we can decompose the horizontal velocity into $u = \bar{u} + u'$ and $v = \bar{v} + v'$, where $\bar{u}$ and $\bar{v}$ are the time-averaged terms, u' and v' are the fluctuating terms. The decomposition also assumes that $\overline{u'} = 0$ and $\overline{u'\bar{q}} = 0$ for each time step. In other words, the fluctuating terms are sub-grid behavior and has zero average in each grid of time step. In the equations the bar notations in the time-averaged terms are dropped for simplicity.

- The hydrostatic approximation is used, since the horizontal scale is usually larger than the vertical scale. Under this approximation, all the terms including the vertical velocity w are ignored except for the turbulent term.

- The Boussinesq approximation is applied, which is valid when the temperature varies little in the model. Using this approximation, the density is assumed to be constant except in the buoyancy term $-\frac{\rho g}{\rho_o}$.

- Turbulence closure. In the equations, there are still terms including the fluctuations, such as $\overline{u'w'}$, resulting in the more unknown variables than the number of equations. Therefore, parameterization is needed for those fluctuating or turbulent terms. A commonly used method to parameterize the turbulent terms is a first order closure that approximate the terms with the gradient of time-averaged variables. For example:

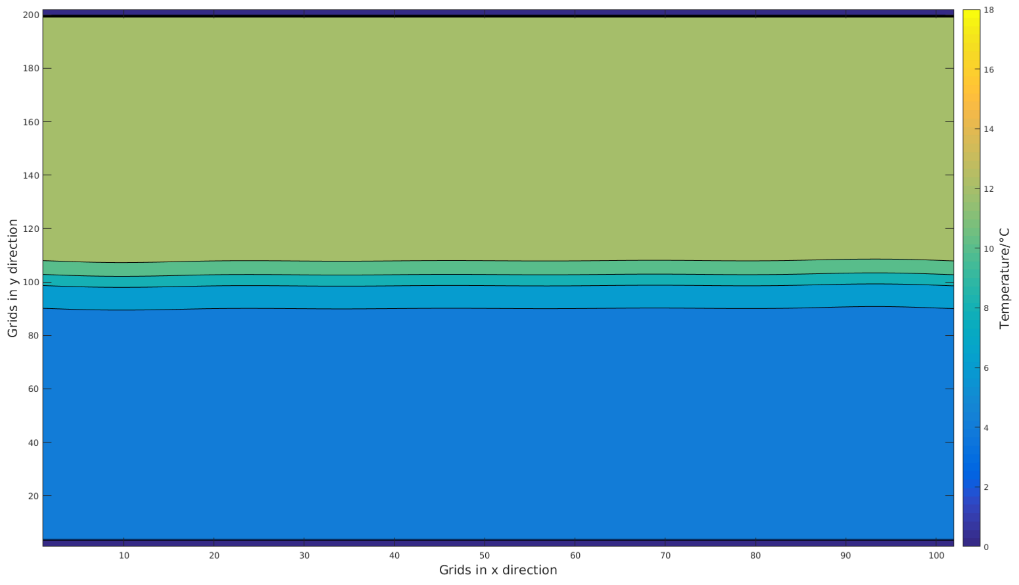
$$\overline{u'w'} = -K_M \frac{\partial u}{\partial z} \tag{5.8}$$

  where K_M is called the eddy diffusivity.

5.2.2 Twin Experiment Setup

The twin experiments we studied with ROMS model are performed on a rectangular region with 200×100 grid points. The initial condition for the actual solution is set to have an temperature gradient and 0 initial velocity. The model is integrated forward for 57600 time steps with a step size of 600s. Then Gaussian noises have been added on temperature and velocities every 144 time steps, which is equal to the time of one day, as the measurements. The initial conditions of temperature and horizontal velocities for both the actual solutions and the measurements are plotted in Fig. 5.7 - Fig. 5.9.

The model uses periodic boundary conditions on left and right sides, and closed boundary



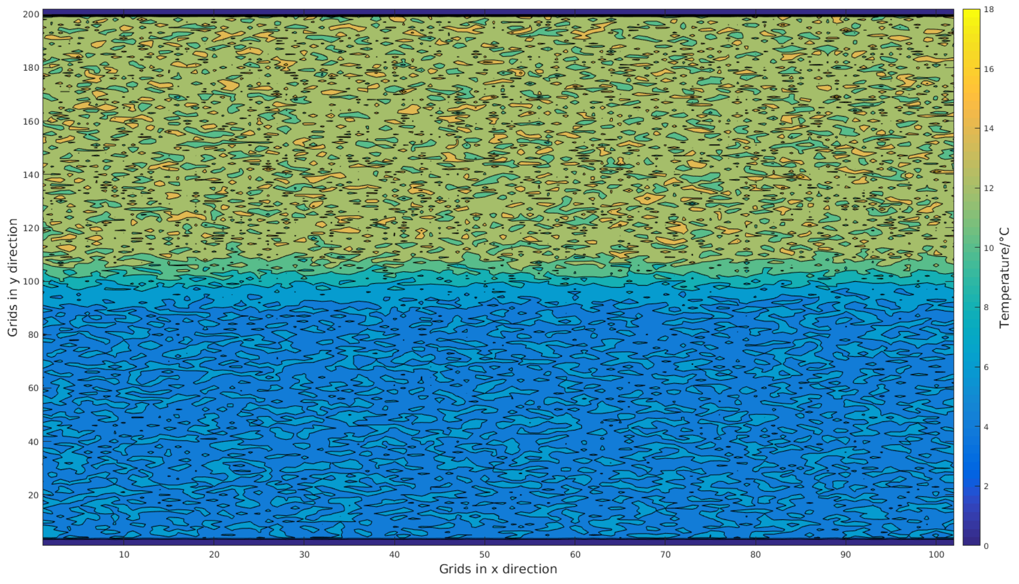
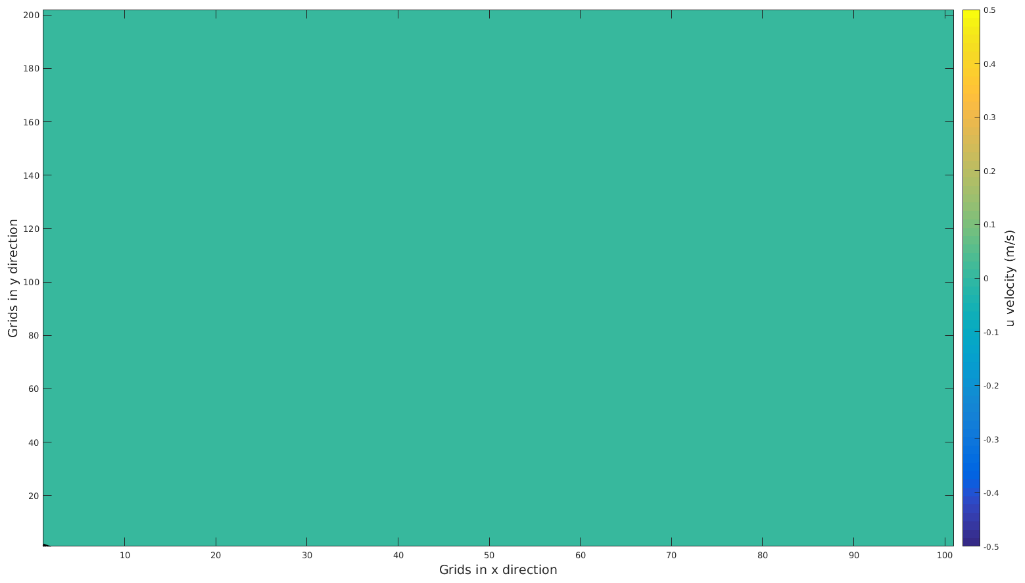


Figure 5.7: Initial conditions of temperatures in ROMS twin experiments setup. Top: The initial condition of the actual temperatures with a gradient in the y direction. Bottom: the measurements with Gaussian noise added on the actual values.



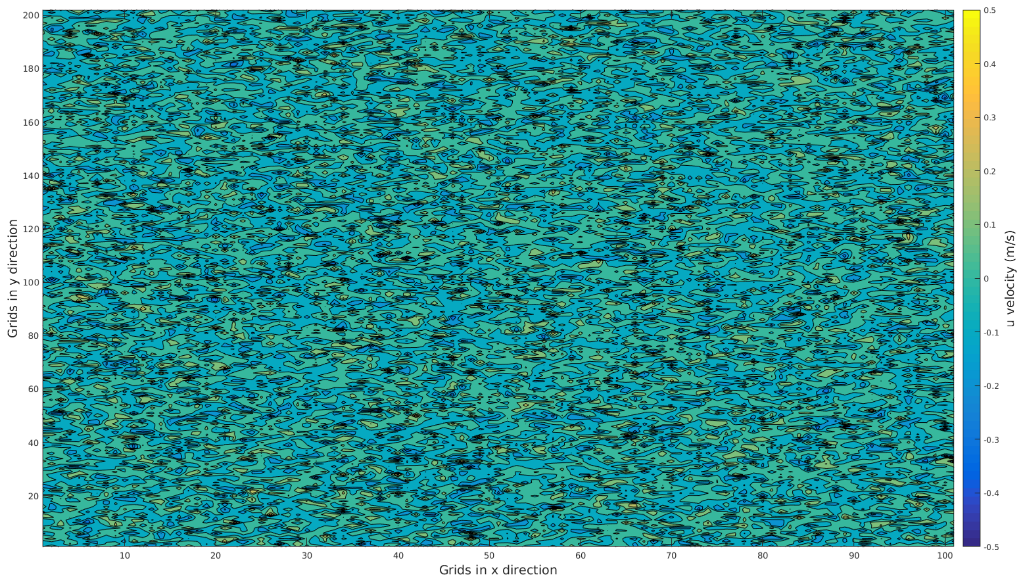


Figure 5.8: Initial conditions of velocities in x direction in ROMS twin experiments setup. Top: The initial condition of the actual velocities which are zero everywhere. Bottom: the measurements with Gaussian noise added on the actual values.

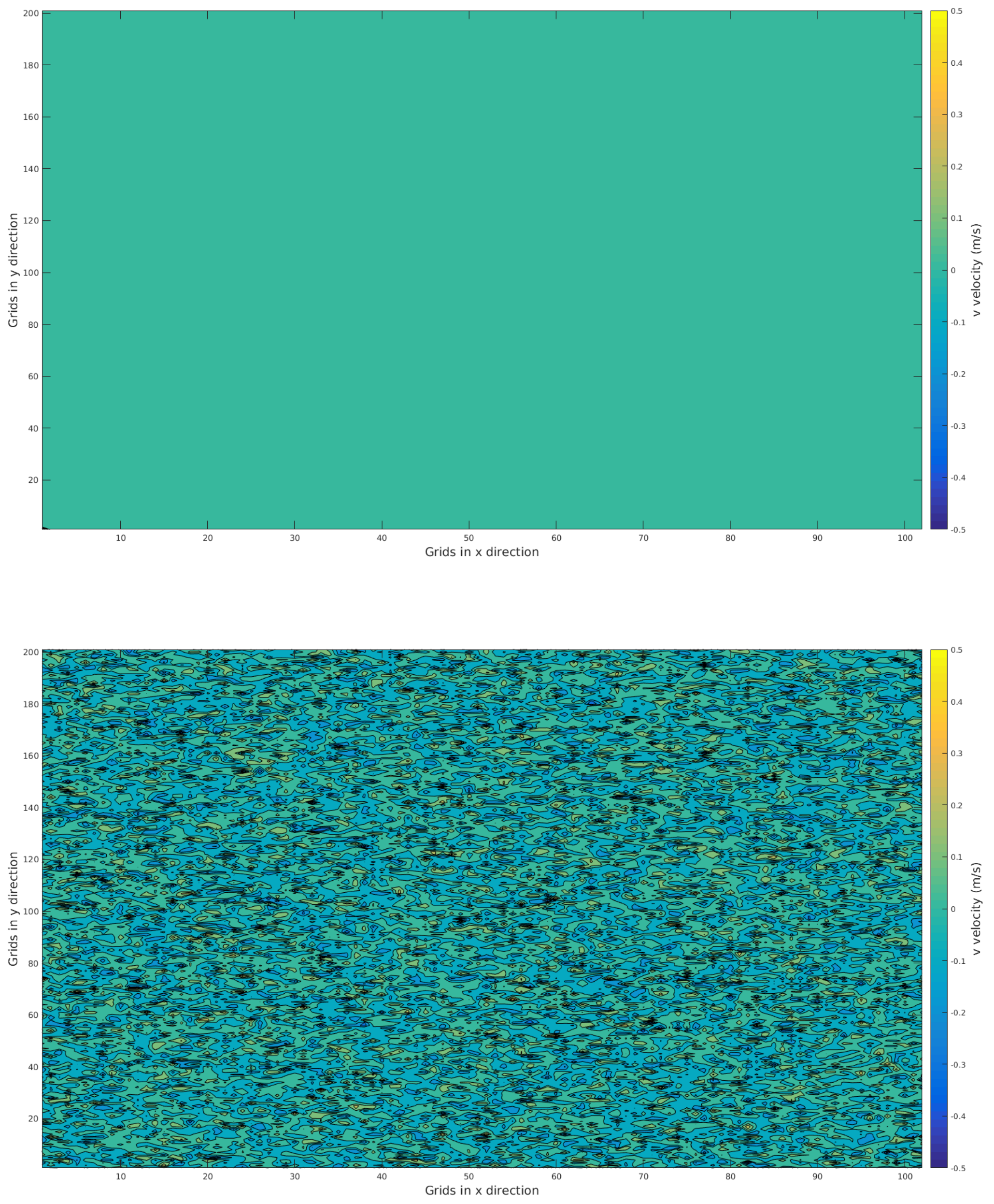


Figure 5.9: Initial conditions of velocities in y direction in ROMS twin experiments setup. Top: The initial condition of the actual velocities which are zero everywhere. Bottom: the measurements with Gaussian noise added on the actual values.

conditions on top and bottom sides in the horizontal direction. No-slip boundary condition is used at the bottom, assuming a layer of constant Reynolds stress near the bottom. The vertical coordinates uses vertical S-coordinate instead of uniformly spaced coordinate points. There are several choices of vertical S-coordinate in the ROMS model. We chose to use the default option, the detail of which can be found in [9].

In addition, a wind force in x direction is added with the form $\tau_u(i,j) = -0.1cos(2\pi y(i,j)/L)m^2/s^2$, where i, j are the coordinates in x, y direction. Table 5.4 summarizes all the configurations of the twin experiments.

Table 5.4: A table to summarize the configurations of the twin experiments on ROMS.

Parameters	Value	Description
N_i	200	Number of grid points in x direction
N_j	200	Number of grid points in y direction
N_σ	20	Number of vertical layers.
dt	600s (10min)	Step size in time
N_t	57600 (400 days)	Number of time steps
N_{ob}	144 (1 day)	Number of time steps between observations
Zo_b	0.02m	Bottom roughness
θ_s	7	Parameter for vertical S-coordinate
θ_b	0.1	Parameter for vertical S-coordinate

In this experiment, the number of variables is significantly larger than previous Lorenz 96 model and shallow water model. In ROMS model, the state variables are: horizontal velocities u and v, temperature T, and surface elevation ζ. So the number of model dimension D is $N_i \times n_j \times (3N_\sigma + 1) \sim 10^6$. Considering the run time of HMC on shallow water model, it would be impractical to apply HMC or PAMC on ROMS model, as it would take too long to finish. Therefore, the following results on ROMS model are based on the nudging method, which is described in section 2.1 of Chapter 2.

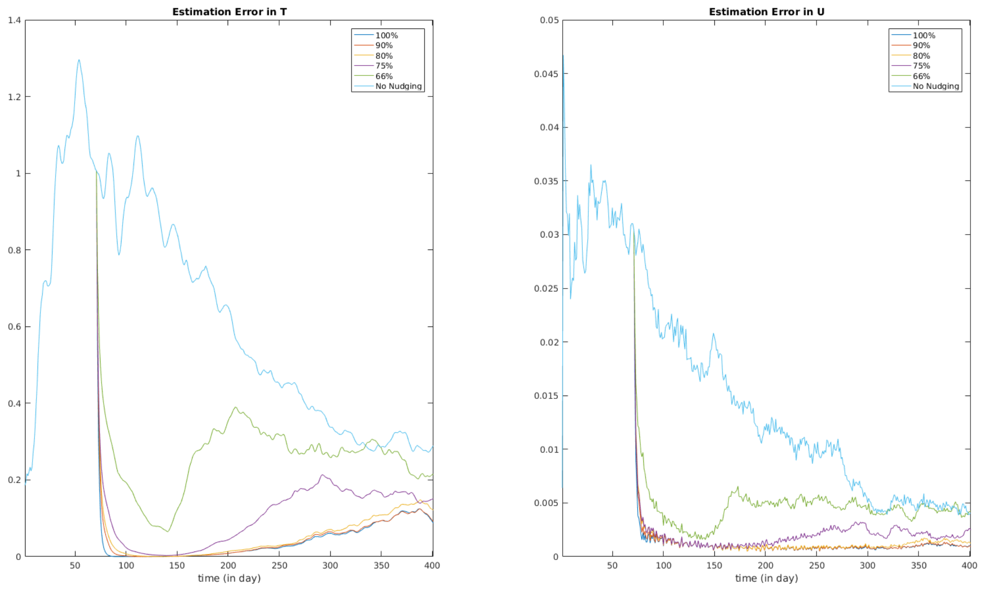


Figure 5.10: Nudging results on ROMS model with different number of measurements. Estimation errors in temperature T and horizontal velocity u with 100%, 90%, 80%, 75%, 66% of variables observed, as well as the error without nudging are plotted.

5.2.3 Results Using the Nudging Method

To study the properties of ROMS model, we did twin experiments on a time window of 400 days. Nudging is applied on the model from day 71 to day 140 using different number of measurements, ranging from 100% to 66% of the total number of variables. We use the root mean square error as the estimation error to quantify the results of nudging method. The errors in temperature T and velocity u over the 400-day period are plotted in Fig. 5.10.

The estimation errors in T and u show similar behavior. As the nudging starts to be used on day 71, the errors drop significantly despite the different number of variables observed. However, when there is only 66% of variables measured, the estimation error is not decreased to the same level when we have more measurements. Then at day 140, when we start the predictions without nudging, the estimation error begin to increase. As shown in the plot, the error grows fastest with only 66% of variables observed, which indicates that the number of measurements is not enough to make good predictions. When we have more than 80$ of variables observed, there is no significant difference in the growth of error, while in the case where we measured 75% of variables, although the estimation error at the end of the nudging window is roughly the same as those of the experiments with more measurements, the error in the prediction window grows faster with only 75% of measurements. Therefore, we know that in this setup of ROMS model, we need about 75% to 80% of measurements in order to successfully synchronize between the model and the measurements and make predictions.

Chapter 6

Conclusions and Future Works

Estimations and predictions may be one of the fundamental and mostly discussed problems people have encountered. The study of data assimilation provides a systematic way to tackle these problems using the noisy, incomplete data, as well as a proposed model, which is most likely imperfect.

In this work, we first described the problems that data assimilation tries to solve, as well as the probabilistic formulation of data assimilation scheme. Then a set of data assimilation methods were discussed, from the nudging method, to two of the recently commonly used methods, the Ensemble Kalman Filter and the 4D-VAR method. We also discussed the precision annealing method in detail, as it will be the main method used in the twin experiment results in this thesis.

In the twin experiment results, we have mainly focus on the problem of the observability, namely how many measurements are required in order to make successful estimations and predictions of a system. We have performed studies on the Lorenz 96 model and the shallow water model, using PAMC and its improvement using HMC respectively. We have found that the Lorenz 96 model needs about 60% measurements to perform well. It is in agreement with previous work [23], which also validates the PAMC method. In Chapter 5, we have also studied a twin experiment setup using ROMS model, which has the number of variables similar to some real applications. Due to the large number of dimension, we have only applied the nudging method to the model, and found out that we need about 75% to 80% measurements to estimate and predict well.

During the implementations of the PAMC method, we have also run into its limitation of computing and memory efficiency. In the originally proposed PAMC method, which uses the Metropolis-Hastings algorithm, the low acceptance rate in high dimensional problems is a significant bottleneck that limits its usage. As a result, the improved method using HMC is developed.

However, the HMC method is also facing the issue of scalability when applying it to some more complicated models such as the shallow water model. In the actual implementation, we have used the autograd package in PyTorch to compute the gradient of action with respect to the state variables and parameters. In this case, even if the GPU is used to parallelize the computation, it still requires more than 5 days to finish the whole annealing method. Furthermore, after tracking the time elapsed at each stage, we found that more than 70% of the time were spent in calculating the gradient. Therefore, some future investigation in the improvement of the method is needed in order to make it capable of large scale problems.

One of the obvious direction is to find a faster method of calculating the gradient. One way is to calculate the form of gradient manually and hand code it into the program, then it would require much less time to evaluate. While this may work in some cases, there are also many models which are hard to calculate the gradient of action due to the complexity of the dynamical equations. So this solution could only have limited usage.

Another option is to find out an alternative formulation of the annealing method, so that it scales better with the increase in the number of state variables and time steps. An example of this is discussed in Chapter 2, which is the development of strong constrained 4D-VAR from weak constrained 4D-VAR method. In this example, the non-linearity of the action or cost function does increase even though the memory and computing time required decrease. However, with the annealing method, the non-linearity may not be a serious drawback, and we can still find the global minimum of the action in many situations.

Appendix A

Details in the Shallow Water Model Implementation

As we have mentioned in Chapter 5, the integration and grid point setup used for the shallow water model follows [26] with a few extra terms such as bottom friction. Here the details of the implementation will be discussed.

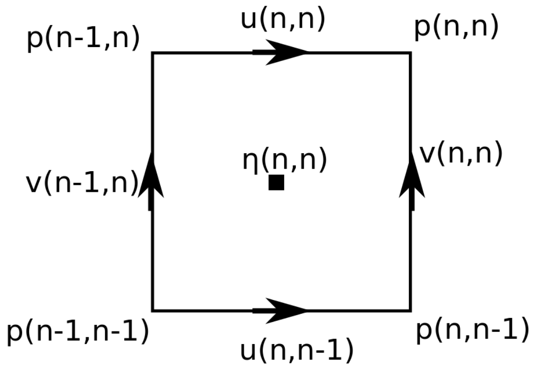


Figure A.1: Staggered grid points for shallow water model.

A.1 Staggered Grid Points

The staggered grid points used in shown in Fig. A.1. For all the state variables, the horizontal velocities, u and v, are defined at the north-south and east-west edges of the grid respectively. The pressure P is defined at the 4 corners. The center of the grid is where an intermediate variable, potential vorticity η, is located at.

A.2 Average and Derivatives

Since now the variables are located at different positions of the grid, in the dynamical equations, the average of a variable needs to be calculated if it is defined at a different location from the other variables included. For example, if the calculation of $\eta(n,n)$ includes the horizontal velocity u and pressure p, then we need to use their averaged forms:

$$\begin{aligned}\overline{u} &= \frac{u(n,n)+u(n,n-1)}{2}\\ \overline{P} &= \frac{P(n,n)+P(n-1,n)+P(n,n-1)+P(n-1,n-1)}{4}\end{aligned} \tag{A.1}$$

Similarly, taking the horizontal velocity u as an example, the derivatives can be written as

$$\frac{\partial u}{\partial x} = \frac{u(n,n)-u(n,n-1)}{dx} \tag{A.2}$$

where dx is the length of the grid in north-south direction.

A.3 Dynamical Equations

First, we need to calculate a few intermediate variables: the mass fluxes U and V in both horizontal directions, a pressure related quantity H, as well as the potential vorticity η.

$$
\begin{aligned}
U &= Pu \\
V &= Pv \\
H &= P + \frac{1}{2}(u^2 + v^2) \\
\eta &= (\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})/P
\end{aligned}
\tag{A.3}
$$

In the equations above and all other equations following in this section, we have omitted the average sign for simplicity, which is needed when variables defined at different positions are included.

Then, the dynamical equations for state variables, u, v and P can be written as follows:

$$
\begin{aligned}
\frac{\partial u}{\partial t} &= \eta V - \frac{\partial H}{\partial x} + \nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) - \varepsilon u + (f_0 + \beta y)v + F \\
\frac{\partial v}{\partial t} &= -\eta U - \frac{\partial H}{\partial y} + \nu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) - \varepsilon v - (f_0 + \beta y)u \\
\frac{\partial P}{\partial t} &= -\frac{\partial U}{\partial x} - \frac{\partial V}{\partial y}
\end{aligned}
\tag{A.4}
$$

where we added a few extra terms same as in Eq. 5.1 and 5.2, except that F is used to represent the surface wind stress term instead of $-\hat{z} \cdot \nabla \times [\frac{\tau(\vec{r})}{\rho f(\vec{r})}]$ for convenience.

Bibliography

[1] H. Abarbanel. *Predicting the future: completing models of observed complex systems*. Springer, 2013.

[2] M. Betancourt. The convergence of markov chain monte carlo methods: from the metropolis method to hamiltonian monte carlo. *Annalen der Physik*, 531(3):1700214, 2019.

[3] G. P. Cressman. An operational objective analysis system. *Mon. Wea. Rev*, 87(10):367–374, 1959.

[4] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[5] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

[6] G. Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

[7] R. M. Fano and D. Hawkins. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29:793–794, 1961.

[8] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[9] K. Hedstrom. Technical manual for a coupled sea-ice/ocean circulation model (version 4). us dept. of the interior, bureau of ocean energy management, alaska ocs region. ocs study boem 2016-037. 176 pp. *This document was prepared with LATEX xfig, and inkscape*, page 4, 2016.

[10] J. E. Hoke and R. A. Anthes. The initialization of numerical models by a dynamic-initialization technique. *Monthly Weather Review*, 104(12):1551–1556, 1976.

[11] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[12] E. Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

[13] M. Kostuk, B. A. Toth, C. D. Meliza, D. Margoliash, and H. D. Abarbanel. Dynamical estimation of neuron and network properties ii: path integral monte carlo methods. *Biological cybernetics*, 106(3):155–167, 2012.

[14] P. S. Laplace. Memoir on the probability of the causes of events. *Statistical Science*, 1(3):364–378, 1986.

[15] B. Leimkuhler and S. Reich. *Simulating hamiltonian dynamics*, volume 14. Cambridge university press, 2004.

[16] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[17] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[18] E. N. Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.

[19] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[20] A. Miller, D. Li, J. Platt, A. Daou, D. Margoliash, and H. Abarbanel. Statistical data assimilation: Formulation and examples from neurobiology. *arXiv preprint arXiv:1809.05196*, 2018.

[21] A. M. Moore, H. G. Arango, G. Broquet, B. S. Powell, A. T. Weaver, and J. Zavala-Garay. The regional ocean modeling system (roms) 4-dimensional variational data assimilation systems: Part i–system overview and formulation. *Progress in Oceanography*, 91(1):34–49, 2011.

[22] R. M. Neal. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[23] J. C. Quinn. *A path integral approach to data assimilation in stochastic nonlinear systems*. PhD thesis, UC San Diego, 2010.

[24] F. Rabier, H. Järvinen, E. Klinker, J.-F. Mahfouf, and A. Simmons. The ecmwf operational implementation of four-dimensional variational assimilation. i: Experimental results with simplified physics. *Quarterly Journal of the Royal Meteorological Society*, 126(564):1143–1170, 2000.

[25] O. Reynolds. Iv. on the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical transactions of the royal society of london.(a.)*, (186):123–164, 1895.

[26] R. Sadourny. The dynamics of finite-difference models of the shallow-water equations. *Journal of the Atmospheric Sciences*, 32(4):680–689, 1975.

[27] C. Sparrow. *The Lorenz equations: bifurcations, chaos, and strange attractors*, volume 41. Springer Science & Business Media, 2012.

[28] S. H. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC Press, 2018.

[29] O. Talagrand. Assimilation of observations, an introduction (gtspecial issueltdata assimilation in meteology and oceanography: Theory and practice). *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):191–209, 1997.

[30] W. G. Whartenby, J. C. Quinn, and H. D. Abarbanel. The number of required observations in data assimilation for a shallow-water flow. *Monthly Weather Review*, 141(7):2502–2518, 2013.

[31] J. Ye, N. Kadakia, P. Rozdeba, H. Abarbanel, and J. Quinn. Improved variational methods in statistical data assimilation. *Nonlinear Processes in Geophysics*, 22(2):205–213, 2015.

[32] J. Ye, D. Rey, N. Kadakia, M. Eldridge, U. I. Morone, P. Rozdeba, H. D. Abarbanel, and J. C. Quinn. Systematic variational method for statistical nonlinear state and parameter estimation. *Physical Review E*, 92(5):052901, 2015.

[33] G. Zhong and J. E. Marsden. Lie-poisson hamilton-jacobi theory and lie-poisson integrators. *Physics Letters A*, 133(3):134–139, 1988.

[34] E. Ziegel. Numerical recipes: The art of scientific computing, 1987.