UNIVERSITY OF CALIFORNIA SAN DIEGO

Methods for Inference and Prediction in Nonlinear Dynamical Systems

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

in

Physics

by

Zheng Fang

Committee in charge:

Professor Henry D. I. Abarbanel, Chair
Professor Gert Cauwenberghs
Professor Kenneth A. Intriligator
Professor Gabriel A. Silva
Professor Congjun Wu

2021

The dissertation of Zheng Fang is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2021

DEDICATION

To my family.

EPIGRAPH

All models are wrong,
but some are useful.
—George E. P. Box

To unify knowing and acting.
—Wang Yangming (1472-1529)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

It is hard to believe that a pleasant journey through which I am indebted to many is about to conclude.

My first thanks go to my advisor, Professor Henry Abarbanel, who is in many aspects a role model to me. His dedication to physics has always been an inspiration; his guidance and support made me a scientist, and his knowledge and kindness showed me what it is like to be an amazing mentor.

I would like to thank Professors Gert Cauwenberghs, Kenneth Intriligator, Gabriel Silva, and Congjun Wu of my dissertation committee for their valuable time and suggestions.

I would also like to thank Zhe An, Kangbo Hao, Nirag Kadakia, Anna Miller, Jason Platt, Paul Rozdeba, Sasha Shirman, Adrian Wong, and other colleagues in the research group for upholding a stimulating environment for doing science.

I want to thank all my friends at UCSD for making graduate school more colorful.

I thank Sharmila Poddar and Catherine McConney for all their help.

Finally, I owe countless thanks to my grandparents, parents, and Nan for their unwavering support.

This dissertation contains material as it appears in Zheng Fang, Adrian S. Wong, Kangbo Hao, Alexander J. A. Ty, and Henry D. I. Abarbanel: Precision annealing Monte Carlo methods for statistical data assimilation and machine learning, *Physical Review Research*, 2(1), 2020. The dissertation author was the primary investigator and author of this paper.

VITA

2014	Bachelor of Science, University of Science and Technology of China
2014	Visiting Researcher, Brown University
2021	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Zheng Fang, Adrian S. Wong, Kangbo Hao, Alexander J. A. Ty, and Henry D. I. Abarbanel. Precision annealing Monte Carlo methods for statistical data assimilation and machine learning. *Physical Review Research*, 2(1):013050, 2020.

Alexander J. A. Ty, Zheng Fang, Rivver A. Gonzalez, Paul. J. Rozdeba, and Henry D. I. Abarbanel. Machine learning of time series using time-delay embedding and precision annealing. *Neural Computation*, 31(10):2004–2024, 2019.

ABSTRACT OF THE DISSERTATION

Methods for Inference and Prediction in Nonlinear Dynamical Systems

by

Zheng Fang

Doctor of Philosophy in Physics

University of California San Diego, 2021

Professor Henry D. I. Abarbanel, Chair

Transferring information from data to models is crucial to many scientific disciplines. Typically, the data collected are noisy, and the total number of degrees of freedom of the model far exceeds that of the data. For data assimilation in which a physical dynamical system is of interest, one could usually observe only a subset of the vector state of the system at any given time. For an artificial neural network that may be formulated as a dynamical model, observations are limited to only the input and output layers; the network topology of the hidden layers remains flexible. As a result, to train such dynamical models, it is necessary to simultaneously estimate both the observed and unobserved degrees of freedom in the models, along with all the time-independent parameters. These requirements

bring significant challenges to the task.

This dissertation develops methods for systematically transferring information from noisy, partial data into nonlinear dynamical models. A theoretical basis for all these methods is first formulated. Specifically, a high-dimensional probability distribution containing the structure of the dynamics and the data is derived. The task can then be formally cast as the evaluation of an expected-value integral under that probability distribution. A well-studied sampling procedure called Hamiltonian Monte Carlo is then introduced as a functioning part to be combined with Precision Annealing, a framework for gradually enforcing the model constraints into the probability distribution.

Numerical applications are then demonstrated on two physical dynamical systems. In each case, inferences are made for both the model states within the observation window and the time-independent parameters. Once complete, the predictive power of the model is then validated by additional data. Following these is a discussion of the role of the state-space representation.

The dissertation concludes with an exploration of new methods for training artificial neural networks without using the well-known backpropagation procedure. Given the equivalence between the structure of an artificial neural network and that of a dynamical system, the aforementioned theoretical basis is applicable in this arena. The computational results presented indicate promising potentials of the proposed methods.

Chapter 1

Introduction

An important question that remains central to many scientific disciplines is how to transfer information from data to models effectively. In a broad sense, data are a collection of observations that are not only noisy but also almost certainly incapable of representing all the causal factors that produced them. A model can be constructed with physical or statistical principles, or a combination thereof. To physicists, a model often refers to a *dynamical system* that describes the evolution of certain state variables. When the evolution mechanism itself is time-independent, the model usually takes the form of a set of autonomous differential equations. Instead, when the system is open to or driven by external signals, the differential-equation representation is still valid, but the model equations are no longer autonomous. On the other hand, to statisticians, a model is usually a data-trained function that can produce an informed response when given input from the same distribution as the training data. Balancing the trade-off between the bias and the variance of a model is an eternal task when the model is non-physical so that its flexibility needs to be adjusted for optimal performance.

Regardless of how the model is constructed, a common goal of all forms of data analysis involves completing the model using information extracted from observations.

Upon a successful transfer of information, the completed model then serves as an inferential or predictive device that can be put into production when needed.

This dissertation proposes *methods* that can accomplish the goal of information transfer for a wide range of data sets and models. We first discuss the theory behind all these methods as well as why they are well suited for tackling the typical challenges that could arise. We then study the applications of the proposed methods in both physical dynamical systems and artificial neural networks that can be formulated as dynamical systems.

1.1 The Information Transfer Problem

As noted above, bridging observations with a physical or statistical model is a highly nontrivial task. For data, they are noisy and incomplete. For models, we can never be sure of them being precise descriptions of the data-generating dynamics, and sometimes we do not even know how flexible they should be.

For example, in numerical weather prediction, one has recordings of meteorological variables such as air pressure and temperature at several locations for some time. The model consists of a set of fluid-dynamics equations with the number of degrees of freedom far exceeding the size of the observations. The reason for the difference between the size of the model and that of the observations is that the model needs to include variables that cannot be directly observed, such as winds and temperatures aloft at specific altitudes. The task, then, is to estimate all the observed and unobserved degrees of freedom of the model along with all the time-independent model parameters. Once done, the completed model may be used for weather forecasting.

Another example is machine learning. Most models that belong to this category do not rely on physical laws, but they allow greater flexibility in the formulation and often

require a procedure called model selection. In linear regression, one is free to introduce as many feature variables as needed to interpret the data. However, either too few or too many features will lead to poor model performance. In training artificial neural networks for classification, one can only "observe" the training samples at the input and output layers. For hidden layers, the network topology remains flexible, and the states and the inter-layer weights are to be determined by the data.

Generally, the type of information transfer that involves physical dynamical models is called data assimilation [1, 2, 3, 4, 5, 6], as described in the first example above. In the second example, artificial neural networks do not represent physical processes; nonetheless, they too demand information be transferred from data [7, 8, 9, 10, 11, 12, 13, 14].

These two seemingly distinct challenges, namely data assimilation and deep learning, have been shown to be equivalent in their underlying structure [15]. In artificial neural networks, the activation functions that direct the states of nodes from layer to layer are equivalent to the rules that govern the temporal evolution of dynamical variables in physical systems. This crucial equivalence enables us to study the theory of information transfer that covers both data assimilation and deep learning. Moreover, we could propose numerical approaches for both fields that share the same theoretical basis.

In this dissertation, we formulate the information transfer problem as the evaluation of an expected-value integral in high-dimensional space. This formulation applies in both data assimilation and deep learning [16, 17]. We also derive numerical approaches from such formulation and apply them in a variety of computational problems.

For the dynamical systems to be discussed in this dissertation, the most general form reads

$$\frac{dx_a(t)}{dt} = \mathcal{F}_a\left(\mathbf{x}(t), t, \boldsymbol{\theta}\right), \qquad a = 1, \ldots, D, \tag{1.1}$$

where $\mathbf{x}(t) = (x_1(t), \ldots, x_D(t))$ is the D-dimensional vector state of the system at time t, $\mathcal{F}_a(\cdot)$ is the rule for advancing in time for the a-th component, and $\boldsymbol{\theta}$ is the collection of

time-independent parameters. Eq. (1.1) can represent both physical systems and artificial neural networks, as we will see in later chapters.

Let us now consider a motivating example. In 1963, Edward Lorenz proposed a dynamical system (hereinafter referred to as Lorenz63) of three ordinary differential equations [18], which is among the simplest representations of a deterministic chaotic flow and has been widely used for testing methods in nonlinear dynamics. The system is an abstraction from the partial differential equations of thermal convection in lower atmosphere derived by Saltzman [19, 20]. The equations for Lorenz63 are

$$\begin{aligned} \frac{dx(t)}{dt} &= \sigma \cdot [y(t) - x(t)], \\ \frac{dy(t)}{dt} &= x(t) \cdot [\rho - z(t)] - y(t), \\ \frac{dz(t)}{dt} &= x(t) \cdot y(t) - \beta z(t), \end{aligned} \tag{1.2}$$

where $x(t)$, $y(t)$, and $z(t)$ are dynamical variables that are the components of the vector state of the system, and σ, ρ, and β are constant parameters. We choose $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$ for this example. At these or nearby values, the system exhibits chaotic behavior, resulting in localized, non-periodic motion in the space spanned by x, y, and z. In general, chaos is irregular in time, and it has structure in phase space [20].

Fig. 1.1(a) shows the trajectory (or "attractor") of the vector state of Lorenz63 as a result of the chosen parameters. Assuming we are standing on the x-axis and can only record a noisy version of $x(t)$ as shown in Fig. 1.1(b), and we do not know anything about the parameters or the other two components as shown in Fig. 1.1(c) or (d), can we reconstruct the attractor by simultaneously estimating *all three* components of the vector state within the observation window $0 \leq t \leq 30$ as well as the parameters σ, ρ, and β? This question is so important that it clearly represents the goal for all the methods and applications to be presented in this dissertation.

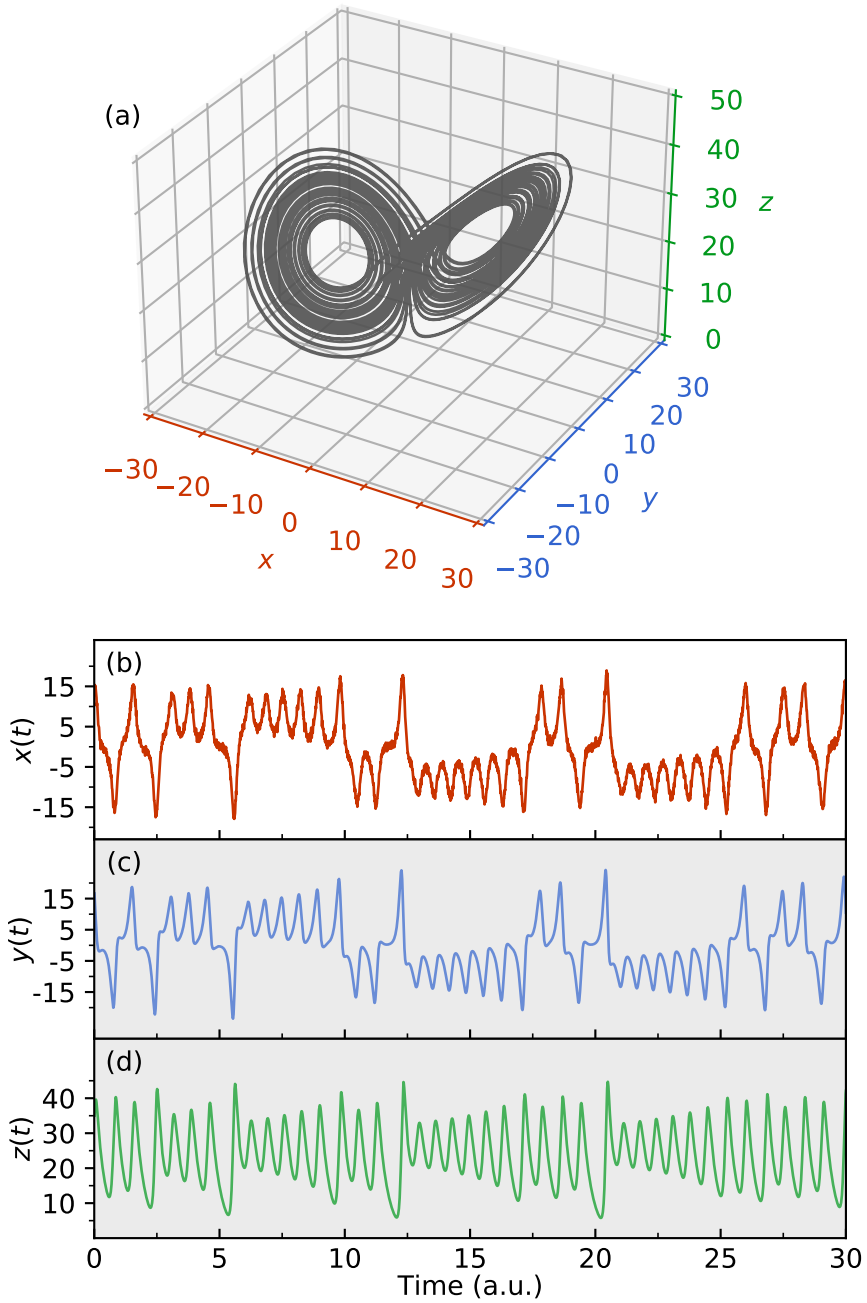

Figure 1.1: Display of the attractor of the Lorenz63 system and the evolution of the three dynamical variables of that system. (a) A plot of the numerical solution to Eq. (1.2) with $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. The discretization interval is $\Delta t = 0.01$, and the number of steps shown is 5000. (b) Dynamical variable $x(t)$ for the first 3000 steps. A Gaussian noise was added. (c) Dynamical variable $y(t)$, which is designated as "unobserved." (d) Dynamical variable $z(t)$, which is designated as "unobserved."

1.2 Example Physical Systems

In this section, we exhibit two physical dynamical systems that are representative of what is generally of interest. One is a biophysical model, and the other is a geophysical model. Although deep learning is another focus of this dissertation, the corresponding discussion is deferred to Chapter 5.

1.2.1 Hodgkin-Huxley Model

We show here a model that is useful for determining the electrophysiological properties of isolated neurons from the HVC nucleus of an avian birdsong system.

The biophysical equations describing the dynamics of neurons were established by work done before and after World War II. Alan Hodgkin and Andrew Huxley [21, 22, 23] are two of the researchers whose names are prominent in understanding that (1) equations imposing current conservation on the ions flowing into and departing from the neuron body and (2) equations capturing the voltage-dependent permeability of the cell membrane to these ions would collectively provide a quantitative framework for describing neuronal dynamics. Fig. 1.2 shows a simulated time series of the transmembrane voltage in response to a stimulating current, $I_{\text{inj}}(t)$. The voltage and the current are the only two pieces of information that can be experimentally observed.

Hodgkin and Huxley considered two ion channels for sodium and potassium ions flowing through the cell membrane. They also introduced a "leak" channel describing other aspects of neuron behavior. The model they proposed have the form

$$\begin{aligned} C\,\frac{dV(t)}{dt} &= g_{\text{Na}}\, m(t)^3\, h(t)\ (E_{\text{Na}} - V(t)) \\ &\quad + g_{\text{K}}\, n(t)^4\ (E_{\text{K}} - V(t)) + g_{\text{L}}\ (E_{\text{L}} - V(t)) \\ &\quad + I_{\text{DC}} + I_{\text{inj}}(t)\,. \end{aligned} \tag{1.3}$$

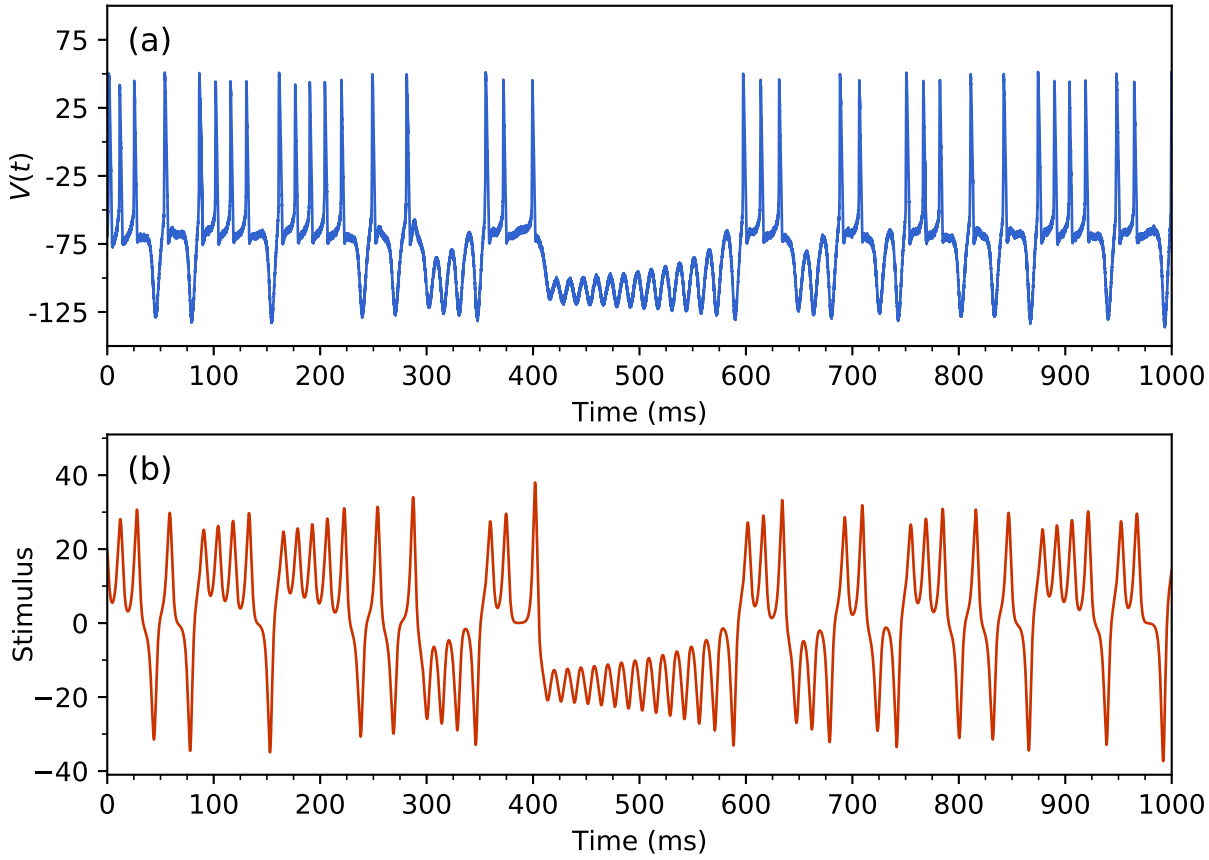

Figure 1.2: The simulated transmembrane voltage in response to a stimulating current. (a) The noisy voltage within the window of $0 \leq t \leq 1000$ms. There any many spikes in the time series. (b) The applied stimulating current, $I_{\text{inj}}(t)$, as an external signal for the Hodgkin-Huxley neuron model.

The three voltage-dependent "gating" variables, i.e., $A_j(t) = m(t), h(t), n(t)$, $0 \leq A_j(t) \leq 1$ are taken to satisfy the first-order kinetics

$$\frac{dA_j(t)}{dt} = \frac{A_{j0}(V(t)) - A_j(t)}{\tau_j(V(t))}\,. \tag{1.4}$$

The parameters $g_{\rm Na}$, $g_{\rm K}$, and $g_{\rm L}$ in the voltage equation, Eq. (1.3), are constants while the gating variables are state variables of the model. $A_{j0}(V(t))$ and $\tau_j(V(t))$ are voltage-dependent functions. The first is dimensionless and sets the scale for the gating variables, and the second is a voltage-dependent timescale for the activity of the gating variables.

The Hodgkin-Huxley model is a dynamical system with rich behavior. In laboratory experiments one can directly measure the transmembrane voltage $V(t)$, but no instruments are available to measure the gating variables. In the general language used throughout this dissertation, this means that the three state variables $A_j(t)$ are *unobserved*.

An experiment consists of selecting a stimulating current $I_{\rm inj}(t)$, which is typically known. With only $V(t)$ observed, the challenge is to estimate all the $A_j(t)$'s in Eq. (1.4) over an observation window $[t_0, t_{\rm final}]$ as well as all the fixed parameters in the two equations above. We also estimate $V(t)$ over this interval as it is always noisy.

Once this is done, the model for the neuron is complete, and with a set of initial conditions $\{V(t_{\rm final}), A_j(t_{\rm final})\}$ at $t = t_{\rm final}$, we may solve this initial value problem for $t \geq t_{\rm final}$. Assessing the quality of the information transfer comes from solving Eqs. (1.3) and (1.4) for $t \geq t_{\rm final}$, and comparing the model output $V(t)$ with observed values in a prediction window. An earlier result that used this model can be found in Ref. [24].

Why should we expect the transfer of information will work? Because the model is nonlinear in the state variables that are generically coupled together through the model. Information is passed from the data through the observed degrees of freedom, and the unobserved ones can be determined with the help of the coupling in the model.

1.2.2 Shallow-Water Equations

Here we briefly discuss estimating the number of required observations in the shallow-water equations, a core component of many numerical weather models.

As the depth of the atmosphere or the ocean fluid layer is markedly less than the earth's radius, the shallow-water equations for two-dimensional flow are an excellent approximation to the fluid dynamics in such a geometry [25, 26]. Three fields on a mid-latitude plane describe the fluid flow: the east-west velocity $u(\mathbf{r}, t)$, the north-south velocity $v(\mathbf{r}, t)$, and the height of the fluid $h(\mathbf{r}, t)$, with $\mathbf{r} = \{x, y\}$. The fluid is taken as a single, constant density layer and is driven by wind stress $\tau(\mathbf{r}, t)$ at the $z = 0$ surface through an Ekman layer. These physical processes satisfy the following dynamical equations with $\mathbf{u}(\mathbf{r}, t) \equiv \{u(\mathbf{r}, t), v(\mathbf{r}, t)\}$:

$$\begin{aligned} \frac{\partial \mathbf{u}(\mathbf{r}, t)}{\partial t} &= -\,\mathbf{u}(\mathbf{r}, t) \cdot \nabla \mathbf{u}(\mathbf{r}, t) - g\,\nabla h(\mathbf{r}, t) \\ &\qquad + \mathbf{u}(\mathbf{r}, t) \times f(y)\hat{\mathbf{z}} + A\,\nabla^2 \mathbf{u}(\mathbf{r}, t) - \varepsilon\,\mathbf{u}(\mathbf{r}, t), \\ \frac{\partial h(\mathbf{r}, t)}{\partial t} &= -\,\nabla \cdot [h(\mathbf{r}, t)\,\mathbf{u}(\mathbf{r}, t)] - \hat{\mathbf{z}} \cdot \operatorname{curl}\left[\frac{\tau(\mathbf{r}, t)}{f(y)}\right]. \end{aligned} \tag{1.5}$$

The Coriolis force is linearized about the equator with $f(y) = f_0 + \beta y$ and the wind-stress profile is selected to be $\tau(\mathbf{r}) = \{F \cos(2\pi y)/\rho\,, 0\}$. The parameter A represents the viscosity in the shallow-water layer, ε is the Rayleigh friction, and $\hat{\mathbf{z}}$ is the unit vector in the z direction. With some chosen parameters, the shallow-water flow is chaotic, and the largest Lyapunov exponent for this flow is $\lambda_{\max} = 0.0325\,\mathrm{h}^{-1} \approx 1/31\,\mathrm{h}^{-1}$.

An earlier work [27] analyzed this flow, using the enstrophy-conserving discretization scheme from Ref. [28], on a periodic grid of size N^2 for $N = 16, 32, 64$ with periodic boundary conditions. In that work, given a grid on which to solve the fluid-dynamical equations, $3N^2$ values of $\{u(x, y, t), v(x, y, t), h(x, y, t)\}$ on the (x, y) grid were generated, and $\Delta t = 36\,\mathrm{s}$ was taken to be the time step for the solution.

In Ref. [27], Whartenby et al. performed a "twin experiment" where they solved Eq. (1.5) on a grid and added noise to each of the $D = 3N^2$ output time series. These were stored away as the data. A subset of these data, an L-dimensional time series, was presented to the model using a "nudging" data assimilation method [29]. They estimated that approximately 70% of the $D = 3N^2$ degrees of freedom must be observed in order to synchronize the model output with the data. Specifically, they looked at $N = 16$ and $L = 524$. In the twin experiments, they knew all $D = 768$ time series on the grid, so they could compare the solutions to Eq. (1.5) to the estimations from the data assimilation procedure. They concluded that for $L < 524$ the estimations differed significantly in the observation window as well as in the prediction window. When $L \geq 524$, this did not happen.

1.3 Outline of the Dissertation

In this chapter, we set the scope of this dissertation. First, we discussed the importance of transferring information from noisy, partial data into dynamical system-like nonlinear models, including physical dynamical systems and artificial neural networks. We previewed the typical challenges and gave a motivating example. We followed up with two physical dynamical systems that exemplify the the data assimilation procedure.

Chapter 2 formulates the theoretical basis of the methods proposed in this dissertation. It begins with introducing a general framework of expressing discretized dynamical models in the state space; a diagram is given in Fig. 2.1. Then, in Sec. 2.2, a high-dimensional probability distribution, denoted by $\pi(\mathbf{X} \,|\, \mathbf{Y})$, that contains the data and the structure of the dynamics of interest is derived. The *action*, $A(\mathbf{X})$, which satisfies $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$, has a convenient form that separates the data with the nonlinear dynamical structure. The problem of information transfer is then cast as the evaluation

of an expected-value integral regarding $A(\mathbf{X})$. After that, Secs. 2.3 and 2.4 assemble the pieces and formally presents the Precision Annealing Hamiltonian Monte Carlo (PAHMC) method that can accomplish the goal of transferring information.

In Chapter 3, we present a series of numerical results that validate the efficacy of the PAHMC method. In Sec. 3.1, the first set of results are obtained on a nonlinear dynamical model widely used in geophysics called Lorenz96. It exhibit chaotic solutions at our choice of the parameter value. Specifically, the model we use has 4001 degrees of freedoms, and we use PAHMC to estimate all of them. We explore several situations in which only a subset of the 4001 degrees of freedom can be observed. The estimation results are then tested by comparing the predictions with data outside the observation window. The second model we study in Sec. 3.2 is the Hodgkin-Huxley model that describes the voltage and other dynamics of a neuron driven by an external current. The model has 20018 degrees of freedom, and about a quarter of that number is observable. Nonetheless, PAHMC gives accurate estimation and prediction results.

Chapter 4 addresses several points related to the theoretical and computational aspects of the proposed methods. In Sec. 4.1, we show that one cannot expect to work only in the initial-condition space and still have a meaningful transfer of information. This point is illustrated in Fig. 4.2. We also show that one also needs to estimate the observed degrees of freedom, despite knowing a noisy version of them. Sec. 4.2 discusses computational efficiency and parallelization of PAHMC. Sec. 4.3 is a summary of the previous chapters.

Chapter 5 proposes an approach for training deep learning models, i.e., artificial neural networks, without involving the widely used backpropagation technique. In Sec. 5.1, we first explain the need for an alternative to backpropagation and then formally describe the proposed approach. Sec. 5.2 demonstrates the advantages of the novel approach through a series of numerical results. The chapter concludes with a discussion of future work.

Chapter 2

A Method for Estimating States and Parameters in Dynamical Models

The focus of this dissertation is how to accurately and efficiently transfer information from noisy, incomplete data into a proprietary dynamical model. As we have seen in Chapter 1, this problem is often referred to as data assimilation [1, 2, 3, 4, 5, 6, 30]. It shares the same goal with many statistical learning procedures: learning from data and making inferences and predictions once the learning is complete. Nonetheless, data assimilation differs from pure statistical learning [31, 32] in many significant ways.

Most importantly, data assimilation assumes an underlying physical dynamical model that explains the time evolution of data. The model can have errors, but it still needs to describe how to drive the dynamics forward in time given information on the past states. In fact, we will see in Sec. 2.2.1 that the Markov property is usually a reasonable assumption and can lead to a nice form of the objective function. As a comparison, statistical learning models, be it parametric or non-parametric, do not make such assumptions. Also, for data assimilation tasks, the data at hand is quite often incomplete, which means we can only observe an L-vector at a given time, out of the total D dimensions ($L < D$) prescribed by

the model.

As a side note, we will use the words "dynamical model" and "dynamical system" interchangeably throughout this dissertation.

This chapter starts with a description of the state-space representation of a dynamical system (in Sec. 2.1) that serves as a stepping stone of all subsequent discussions.

In Sec. 2.2, after observing some mild assumptions, a general form of the objective function for the information transfer task is derived. Specifically, we first derive a quantity called the "action" from a joint probability distribution that contains all the information. We then present the first derivatives of the action, which are very useful in numerical calculations. Following that, we formally state the goal of data assimilation.

Then, in Sec. 2.3, we review the Hamiltonian Monte Carlo (HMC) method, which is an important component of the method proposed in this dissertation. The theoretical foundation of HMC is described, and it is followed by a description of the algorithm itself. This section concludes with a two-dimensional example illustrating the advantages of HMC.

Finally, in Sec. 2.4, we will formally describe the other component of the proposed method, i.e., the Precision Annealing (PA) procedure. Both the motivation and the quantitative formulation will be discussed.

2.1 State-Space Representation

Let us begin with a description of the framework within which the information transfer problem is defined. This section also serves as a survey of notations.

Within an observation window $[t_0, t_{\text{final}}]$, we make a set of measurements at times $t = \{\tau_0, \ldots, \tau_F\}$, where $t_0 \leq \tau_0 < \tau_F \leq t_{\text{final}}$. At each of these measurement times, i.e., $k = 0, \ldots, F$, we observe an L-dimensional vector $\mathbf{y}(\tau_k) = (y_1(\tau_k), \ldots, y_L(\tau_k))$. The number of observations L at each measurement time $t = \tau_k$ is typically less, sometimes much

less, than the number of degrees of freedom D in the dynamical system. That is to say, $L \leq D$. These observations (data) are stored in a library and otherwise unaltered during the remainder of the discussion. We designate the collection of observations made from time $t = \tau_0$ up to $t = \tau_F$ as $\mathbf{Y} \equiv \{\mathbf{y}(\tau_0), \ldots, \mathbf{y}(\tau_F)\}$.

The quantitative characterization of the dynamical system that produced these data is based on our knowledge of the physical dynamics assumed to be descriptive of the data source. If we are performing a task in machine learning, the model may be divorced from knowledge of physics, and it may be instead constructed through statistical principles.

The model describes the interactions among as well as the evolution of the *states* of the target system. From the data, $\mathbf{Y}$, we want to estimate both the *observed* and the *unobserved* states of the model as a function of time. We also need to estimate any time independent parameters in the model. After the information transfer procedure is complete, we will use all the estimated values of model states and parameters to predict the behavior of the dynamical system for $t \geq t_{\text{final}}$. Much like in a standard machine learning task, we can use a validation data set to evaluate the result of the transfer of information by comparing our predictions with what we have in the validation data set.

We now turn to the *states* of the dynamical system, i.e., variables that fully characterize the system at any given time. At time t, we call the D-dimensional state of the model $\mathbf{x}(t) = (x_1(t), \ldots, x_D(t))$. The model is constructed to describe the dynamical behavior of the observations through a set of differential equations in continuous time and is generically written as

$$\frac{dx_a(t)}{dt} = \mathcal{F}_a(\mathbf{x}(t), \boldsymbol{\theta}), \qquad a = 1, \ldots, D. \tag{2.1}$$

Since all numerical calculations happen in discrete time, we must as well structure the model in a discrete fashion. Let us first divide the observation window $[t_0, t_{\text{final}}]$ into

M discrete intervals. That is, in discrete time with an equal interval of Δt, we have $t_m = t_0 + m\Delta t$ and $t_M = t_{\text{final}}$, where $m = 0, \ldots, M$. Under such condition, Eq. (2.1) takes the form $x_a(t_{m+1}) = f_a(\mathbf{x}(t_m), \boldsymbol{\theta})$, or simply

$$x_a(m+1) = f_a(\mathbf{x}(m), \boldsymbol{\theta}), \qquad a = 1, \ldots, D, \tag{2.2}$$

if we define shorthand notation $\mathbf{x}(m) \equiv \mathbf{x}(t_m)$. In Eq. (2.2), $\boldsymbol{\theta} \equiv \{\theta_1, \ldots, \theta_{N_p}\}$ denotes the N_p time-independent parameters intrinsic to the dynamical system. The discrete-time model $f_a(\mathbf{x}(m), \boldsymbol{\theta})$ is related to its continuous counterpart $\mathcal{F}_a(\mathbf{x}(t), \boldsymbol{\theta})$ via the choice for solving the continuous time flow for $\mathbf{x}(t)$ through a discrete time numerical solution method [33].

We work henceforth in discrete time. For convenience, and without loss of generality, we choose the observation times τ_k, $k = 0, \ldots, F$, to be integer multiples of Δt as well. This gives us

$$\tau_k = t_0 + n_k \cdot \Delta t, \qquad k = 0, \ldots, F, \tag{2.3}$$

where $\{n_0, \ldots, n_F\}$ is a set of integers denoting the observation times, and $n_k \leq M$ for $k = 0, \ldots, F$.

Starting from the inception of the observation window at t_0, we must use our model equations to move the state variables $\mathbf{x}(0)$ from t_0 to $\tau_1 = t_0 + n_1 \cdot \Delta t$, where the first measurement is made. Then we use the model dynamics again to move along to $\tau_2 = t_0 + n_2 \cdot \Delta t$, where the second measurement is made, and so forth until we reach the time of the last measurement $\tau_F = t_0 + n_F \cdot \Delta t$ and finally move the model from $\mathbf{x}(\tau_F)$ to $\mathbf{x}(t_{\text{final}}) = \mathbf{x}(M)$.

We collect all the D-vectors $\mathbf{x}(m)$ for $m = 0, \ldots, M$ along with all of the N_p

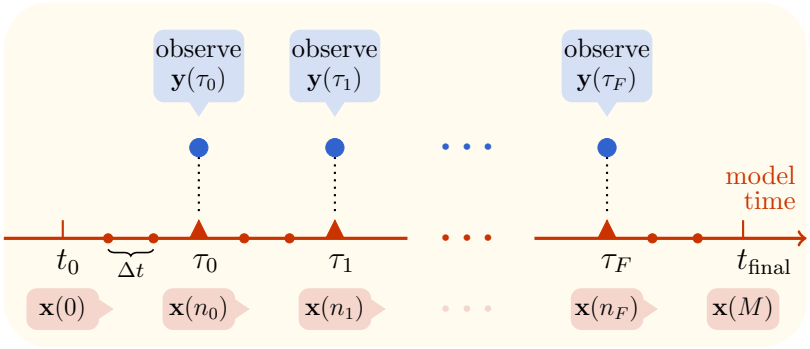

Figure 2.1: A visual representation of the time window $t_0 \leq t \leq t_{\text{final}}$ during which L-dimensional ($L \leq D$) observations $\mathbf{y}(\tau_k)$ are made at times $\tau_k = t_0 + n_k \cdot \Delta t$ for $k = 0, \ldots, F$. The figure also exemplifies time stamps at which the D-dimensional nonlinear model $\mathbf{x}(m+1) = \mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$ is used to move forward from time t_m to t_{m+1} where $t_m = t_0 + m\Delta t$, $m = 0, 1, \ldots, M$, and $t_{\text{final}} = t_M$. In particular, state vectors $\mathbf{x}(n_0)$, $\mathbf{x}(n_1)$, and $\mathbf{x}(n_F)$ have corresponding observations.

parameters into what we call the *path* $\mathbf{X}$, i.e.,

$$\mathbf{X} \equiv \left\{\mathbf{x}(0), \ldots, \mathbf{x}(M); \theta_1, \ldots, \theta_{N_p}\right\}. \tag{2.4}$$

Thus, the dimension of the path $\mathbf{X}$ is $(M+1)D + N_p$.

Fig. 2.1 gives a visualization of the above discussion.

We need three ingredients to accomplish the transfer of information from the collection of all measurements $\mathbf{Y}$ to the model $\mathbf{x}(m+1) = \mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$, along the path $\mathbf{X}$ and through the observation window $[t_0, t_{\text{final}}]$.

(1) The collection of noisy data $\mathbf{Y}$.

(2) A dynamical model describing the generation of $\mathbf{Y}$. This model can be built with physical or other principles.

(3) Methods to achieve the transfer of information from $\mathbf{Y}$ to the model. Note, that the goal is to estimate all the components within $\mathbf{X}$.

Item (3) on the list will be our focus from here on.

In Chapter 3, we will present numerical results using the methods proposed in this

chapter, which then rely on methods of Markov chain Monte Carlo (MCMC) search. Below are the two MCMC methods that are at play.

(1) *The original method* [34, 35]—this starts at some location $\mathbf{X}_i$ in path space, makes proposals to move to a candidate location $\mathbf{X}_c$ by sampling a modification to $\mathbf{X}_i$ drawn from a symmetric distribution. It then accepts or rejects $\mathbf{X}_c$ using the Metropolis-Hastings rule. This procedure is also briefly mentioned in Sec. 2.3. Under some conditions [33, 36], this will explore the target distribution $\pi(\mathbf{X} \,|\, \mathbf{Y})$ well and allow the approximate evaluation of expected values of functions $\langle G(\mathbf{X}) \rangle = \int d\mathbf{X}\, G(\mathbf{X})\, \pi(\mathbf{X} \,|\, \mathbf{Y})$. We label this very well explored construction as the "random proposal" (RP) Monte Carlo method.

(2) *The Hamiltonian Monte Carlo method* [37, 38, 39]. HMC takes off in another direction. It proposes moves from $\mathbf{X}_i$ in an enlarged space reached by adding canonical momenta $\mathbf{P}$ to $\mathbf{X}$. The proposals are made using Hamilton's equations of classical mechanics. HMC will be discussed in Sec. 2.3. A numerical example is given in Sec. 2.3.3. The computational complexity of HMC is discussed in Chapter 4.

Although we discuss the results based on both methods, the plan is to focus more on item (2), i.e., the HMC method.

If the transfer of information is successful and, according to some metric of success, we arrange matters so that at the measurement times τ_k, the L model variables $\mathbf{x}(t = \tau_k)$ associated with $\mathbf{y}(\tau_k)$ are approximately equal to each other, we are *not* finished yet.

At this point, we have only demonstrated that the model is consistent with the known data $\mathbf{Y}$. We must further use the model, completed by the estimates of $\boldsymbol{\theta}$ and the full state of the model at the final time $\mathbf{x}(M)$, to predict forward for $t > t_M$, and we should succeed in comparison with measurements for $\mathbf{y}(\tau)$ for $\tau > t_M$. As a measure of success, we may use the same metric as used in the observation window. No further information from the observations is passed to the model in the prediction window.

As noted before, the same overall setup also applies to artificial neural networks in supervised machine learning [8, 9], where the training set is used to transfer information into the model (by adjusting the weights), the test set is used to make predictions and validate the model. The process of prediction is called generalization.

2.2 The Action and Expected-Value Integrals

It is inevitable that the data at hand are noisy, and the model we construct may not be fully reflective of the "true" dynamics that produced the data. As a result, the transfer of information must be done in a probabilistic manner. In other words, we must study the conditional probability distribution $\pi(\mathbf{X} \,|\, \mathbf{Y})$ as a pivotal quantity for transferring information from $\mathbf{Y}$ to the model $\mathbf{x}(m+1) = \mathbf{f}\,(\mathbf{x}(m), \boldsymbol{\theta})$.

This section first handles the derivation of the negative log-likelihood of $\pi(\mathbf{X} \,|\, \mathbf{Y})$ given mild assumptions. We will then present several equations that are handy for large-scale numerical calculations. The section concludes with identifying the goals and challenges of performing the information transfer task.

2.2.1 Derivation of the Action

We call the negative log-likelihood of $\pi(\mathbf{X} \,|\, \mathbf{Y})$ *action*, and it is denoted by $A(\mathbf{X})$. Quantitatively speaking, we have

$$A(\mathbf{X}) = -\log\left[\pi(\mathbf{X} \,|\, \mathbf{Y})\right] - \log C, \tag{2.5}$$

where C is the unimportant normalizing constant of the conditional distribution. In other words, all we need is the relation $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$.

The derivation now begins with a detailed analysis of the structure of $\pi(\mathbf{X} \,|\, \mathbf{Y})$.

Assume for the moment that we have an equal number of observation and model time steps, so $n_k = k$ in Eq. (2.3) for $k = 0, \ldots, M$. Let us first introduce the shorthand notation: $\mathbf{Y}_0^k \equiv \{\mathbf{y}(0), \ldots, \mathbf{y}(k)\}$ and $\mathbf{X}_0^k \equiv \{\mathbf{x}(0), \ldots, \mathbf{x}(k); \boldsymbol{\theta}\}$ for $k \leq M$.

From $t = \tau_k$ to $t = \tau_{k+1}$ we have a recursion relation relating $\pi(\mathbf{X}_0^{k+1} \,|\, \mathbf{Y}_0^{k+1})$ to $\pi(\mathbf{X}_0^k \,|\, \mathbf{Y}_0^k)$:

$$\begin{aligned}
\pi\left(\mathbf{X}_0^{k+1} \,\middle|\, \mathbf{Y}_0^{k+1}\right) &= \frac{\pi\left(\mathbf{x}(k+1),\, \mathbf{y}(k+1),\, \mathbf{X}_0^k,\, \mathbf{Y}_0^k\right)}{\pi\left(\mathbf{y}(k+1),\, \mathbf{Y}_0^k\right) \cdot \pi\left(\mathbf{x}(k+1),\, \mathbf{X}_0^k,\, \mathbf{Y}_0^k\right)} \\
&\quad \cdot \pi\left(\mathbf{x}(k+1) \,|\, \mathbf{x}(k)\right) \cdot \pi\left(\mathbf{X}_0^k,\, \mathbf{Y}_0^k\right) \\
&= \frac{\pi\left(\mathbf{y}(k+1),\, \mathbf{X}_0^{k+1} \,\middle|\, \mathbf{Y}_0^k\right)}{\pi\left(\mathbf{y}(k+1) \,\middle|\, \mathbf{Y}_0^k\right) \cdot \pi\left(\mathbf{X}_0^{k+1} \,\middle|\, \mathbf{Y}_0^k\right)} \\
&\quad \cdot \pi\left(\mathbf{x}(k+1) \,|\, \mathbf{x}(k)\right) \cdot \pi\left(\mathbf{X}_0^k \,\middle|\, \mathbf{Y}_0^k\right) \\
&= \frac{\pi\left(\mathbf{y}(k+1) \,\middle|\, \mathbf{x}(k+1),\, \mathbf{X}_0^k, \mathbf{Y}_0^k\right)}{\pi\left(\mathbf{y}(k+1) \,\middle|\, \mathbf{Y}_0^k\right)} \\
&\quad \cdot \pi\left(\mathbf{x}(k+1) \,|\, \mathbf{x}(k)\right) \cdot \pi\left(\mathbf{X}_0^k \,\middle|\, \mathbf{Y}_0^k\right).
\end{aligned} \tag{2.6}$$

In establishing the first equal sign in Eq. (2.6), we have made a mild but critical assumption, which is the Markov property of the dynamics $\mathbf{x}(m+1) = \mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$. It can be formally written as

$$\pi\left(\mathbf{x}(k+1) \,\middle|\, \mathbf{X}_0^k, \mathbf{Y}_0^k\right) = \pi\left(\mathbf{x}(k+1) \,|\, \mathbf{x}(k)\right). \tag{2.7}$$

It is implied that Eq. (2.7) holds throughout the discussion. The second and third equal signs in Eq. (2.6) have made use of the basic relation $p(A, B \,|\, C) = p(A \,|\, B, C) \cdot p(B \,|\, C)$.

Before we proceed, it is worth noting that if we start from the right-hand side of

the second equal sign of Eq. (2.6), the equation can also be written as

$$\begin{aligned}\pi\left(\mathbf{X}_0^{k+1} \,\middle|\, \mathbf{Y}_0^{k+1}\right) = \exp\left[\mathrm{CMI}\left(\mathbf{y}(k+1),\, \left\{\mathbf{x}(k+1),\, \mathbf{X}_0^k\right\} \,\middle|\, \mathbf{Y}_0^k\right)\right] \\ \cdot\, \pi\left(\mathbf{x}(k+1) \,|\, \mathbf{x}(k)\right) \cdot \pi\left(\mathbf{X}_0^k \,\middle|\, \mathbf{Y}_0^k\right),\end{aligned} \tag{2.8}$$

where CMI is Shannon's conditional mutual information [40], formally defined as

$$\mathrm{CMI}\left(A, B \,|\, C\right) = \log\left[\frac{\pi\left(A, B \,|\, C\right)}{\pi\left(A \,|\, C\right)\pi\left(B \,|\, C\right)}\right]. \tag{2.9}$$

Essentially, Eq. (2.9) tells us how many bits (when using $\log_2$) we know about A when observing B conditioned on C. For us, $A = \mathbf{y}(k+1)$, $B = \{x(k+1), \mathbf{X}_0^k\}$, and $C = \mathbf{Y}_0^k$. The appearance of the CMI is an explicit indication that information is being transferred from the observations to the model at each measurement time.

Now set aside the condition $n_k = k$, and use Eq. (2.6) to move backwards from the end of the observation window at $t_{\mathrm{final}} = t_0 + M\Delta t$, through the observations at times τ_k, to the start of the window at t_0. Up to factors independent of $\mathbf{X}$, we arrive at

$$\begin{aligned}\pi\left(\mathbf{X} \,|\, \mathbf{Y}\right) \propto \left[\prod_{k=0}^{F} \pi\left(\mathbf{y}(\tau_k) \,\middle|\, \mathbf{x}(\tau_k), \mathbf{Y}_0^{k-1}\right)\right] \\ \times \left[\prod_{m=1}^{M} \pi\left(\mathbf{x}(m) \,|\, \mathbf{x}(m-1)\right)\right] \cdot \pi\left(\mathbf{x}(0)\right),\end{aligned} \tag{2.10}$$

where $\mathbf{Y}_0^{-1}$ denotes the empty set.

As mentioned in the beginning of this subsection, $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$ defines the action, $A(\mathbf{X})$. It contains all the information about the observations and the evolution of the dynamical system, and is thus a complete and sufficient characterization of the information transfer process. To make use of the information contained in the action, it is best that we have at least some ideas about the geometry of $A(\mathbf{X})$. The more we know,

the better the result of the information transfer will be. We will see next that one way to probe into the geometry of the action is to perform certain expected-value integrals.

In practice, when we have an action, a typical quantity of interest is then the expected value of some function of the path $\mathbf{X}$. In other words, we are interested in the expected value of some $G(\mathbf{X})$ given the action. The problem can be formally written as

$$\begin{aligned} \langle G(\mathbf{X}) \rangle &= E\left[G(\mathbf{X}) \,|\, \mathbf{Y}\right] \\ &= \frac{\int d\mathbf{X}\, G(\mathbf{X})\, e^{-A(\mathbf{X})}}{\int d\mathbf{X}\, e^{-A(\mathbf{X})}}\,, \end{aligned} \tag{2.11}$$

where $d\mathbf{X} = \prod_{m=0}^{M} d^D\mathbf{x}(m) \prod_{j=1}^{N_p} d\theta_j$. All factors in the action that are independent of $\mathbf{X}$ are not included. Eq. (2.11) sets the goal for all information transfer problems discussed in this dissertation, but it is often impossible to be evaluated analytically. As a result, we need numerical methods to evaluate it accurately and efficiently. Proposing numerical methods to achieve such a goal is an important topic of this dissertation.

What kinds of $G(\mathbf{X})$ are of interest? A natural choice is the path itself: $G(\mathbf{X}) = \mathbf{X}$. Another could be the covariance matrix of $\mathbf{X}$, if one is interested in the "shape" of $A(\mathbf{X})$ and wants to know the interactions among components of $\mathbf{X}$. Note, that the process of making predictions of the dynamical system for $t > t_{\text{final}}$ also falls under the category of evaluating Eq. (2.11)—all we need is to set $G(\mathbf{X}) = \mathbf{f}\left(\mathbf{x}(M), \boldsymbol{\theta}\right) \equiv \mathbf{f}\left(\mathbf{x}(t_{\text{final}}), \boldsymbol{\theta}\right)$, and then do the forward calculation starting at $t = t_{\text{final}}$ and using a time interval of Δt.

Let us proceed with the derivation of the action. Starting from Eq. (2.10) and noting the definition, Eq. (2.5), we immediately have

$$\begin{aligned} A(\mathbf{X}) = &- \sum_{k=0}^{F} \log\left[\pi\left(\mathbf{y}(\tau_k) \,\middle|\, \mathbf{x}(\tau_k), \mathbf{Y}_0^{k-1}\right)\right] \\ &- \sum_{m=1}^{M} \log\left[\pi\left(\mathbf{x}(m) \,|\, \mathbf{x}(m-1)\right)\right] - \log\left[\pi\left(\mathbf{x}(0)\right)\right]. \end{aligned} \tag{2.12}$$

Eq. (2.12) is intuitive. The first sum represents the information added to $\pi(\mathbf{X} \,|\, \mathbf{Y})$ each time an observation $\mathbf{y}(\tau_k)$ is made. The second sum represents the stochastic transitions of the state variables at each discrete model time.

The last term of Eq. (2.12) comes from the distribution of the initial condition of the model. We often have no knowledge about $\pi(\mathbf{x}(0))$, just as we may have no knowledge about the distribution of the parameters $\boldsymbol{\theta}$. In this situation, we take them to be uniform over the dynamical range of the model variables (or the parameters). They then cancel between the numerator and the denominator in Eq. (2.11). The fact that we do not need to specify a full prior for $\mathbf{X}$—we only need to do it for $\mathbf{x}(0)$—is an important distinction between what we are discussing in this dissertation and classical Bayesian inference [41].

Eq. (2.12) is not final. Recall that we have previously made one assumption about the dynamics $\mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$, we now make two additional assumptions that will collectively pin down a more specific and workable form of the action. Below lists all three assumptions.

(1) The dynamics is Markovian, as denoted by Eq. (2.7).

(2) The observation at time $t = \tau_k$, i.e., $\mathbf{y}(\tau_k)$, is uncorrelated with all previous observations $\mathbf{y}(\tau_0), \ldots, \mathbf{y}(\tau_{k-1})$, $\forall k >= 1$.

(3) The difference between the model state $\mathbf{x}(\tau_k)$ and the observation $\mathbf{y}(\tau_k)$ is a Gaussian random vector with zero mean and a diagonal precision matrix $\mathbf{R}_m = R_m \mathbf{I}$, $\forall k >= 0$; and the difference between two successive model states, $\mathbf{x}(m-1)$ and $\mathbf{x}(m)$, is a Gaussian random vector with zero mean and a diagonal precision matrix $\mathbf{R}_f = R_f \mathbf{I}$, $\forall m >= 1$.

We have already discussed the first assumption. Assumption (2) is mild in the sense that it is a valid approximation for most non-quantum mechanical systems. Assumption (3) is also rather generic and can be replace by other specifications on the observational noise and the errors in the model. The two scalars, R_m and R_f, defined in assumption (3), are important hyper-parameters for numerical calculations (more on them in Chapter 3).

With the three assumptions, Eq. (2.12) further simplifies to what we call the "standard form" of the action, which reads

$$\begin{aligned} A(\mathbf{X}) = & \sum_{k=0}^{F} \sum_{\ell=1}^{L} \frac{R_m}{2(F+1)} \left[x_\ell(\tau_k) - y_\ell(\tau_k) \right]^2 \\ & + \sum_{m=0}^{M-1} \sum_{a=1}^{D} \frac{R_f}{2M} \left[x_a(m+1) - f_a(\mathbf{x}(m), \boldsymbol{\theta}) \right]^2 . \end{aligned} \tag{2.13}$$

We call the first term in Eq. (2.13) *measurement error* and the second term *model error*. Eq. (2.13) is pivotal to all subsequent discussions, and it is the starting point of all numerical calculations regarding the information transfer problem.

One way to explicitly see the balance condition between the measurement error and the model error is to examine the resulting Euler-Lagrange equation when the model time becomes continuous when $\Delta t \to 0$. This is discussed in Ref. [15].

2.2.2 First Partial Derivatives for Numerical Calculations

To probe the geometry of $A(\mathbf{X})$, we should know at least its first partial derivatives. This knowledge is especially important when conducting numerical calculations since the derivatives are required by a broad spectrum of algorithms [33, 38]. This is a standalone subsection aiming at providing the first partial derivatives of $A(\mathbf{X})$ without the dynamics $\mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$ specified. For convenience, and without loss of generality, we assume there is an observation at each model time, i.e., $n_k = k$ in Eq. (2.3) for $k = 0, \ldots, M$.

One good choice for discretizing the original dynamics $\mathcal{F}_a$ is the trapezoidal rule, which reads

$$f_a\left(\mathbf{x}(m), \boldsymbol{\theta}\right) = x_a(m) + \frac{\Delta t}{2} \cdot \left[\mathcal{F}_a(\mathbf{x}(m+1), \boldsymbol{\theta}) + \mathcal{F}_a(\mathbf{x}(m), \boldsymbol{\theta}) \right], \tag{2.14}$$

for $m = 0, \ldots, M-1$.

We define two sets of auxiliary variables as

$$\begin{aligned}\mathcal{J}_{ij}(m,\boldsymbol{\theta}) &= \frac{\partial \mathcal{F}_i(\mathbf{x}(m),\boldsymbol{\theta})}{\partial x_j(m)}\,,\\ \mathcal{G}_{ib}(m,\boldsymbol{\theta}) &= \frac{\partial \mathcal{F}_i(\mathbf{x}(m),\boldsymbol{\theta})}{\partial \theta_b}\,,\end{aligned} \tag{2.15}$$

for $i,j = 1,\ldots,D$, $b = 1,\ldots,N_p$, and $m = 0,\ldots,M$.

Now we can write down the first partial derivatives of $A(\mathbf{X})$ given Eqs. (2.14) and (2.15). Since $A(\mathbf{X})$ is the addition of measurement error and model error terms and the derivatives of the measurement error is trivial, we only give the first partial derivatives of the model error.

For the derivatives with respect to $\mathbf{x}(0),\ldots,\mathbf{x}(M)$, we have, for $a = 1,\ldots,D$,

$$\frac{\partial\,\mathrm{ModErr}(\mathbf{X})}{\partial x_a(m)} = \begin{cases} -\dfrac{R_f(a)}{M+1}\cdot[x_a(1) - f_a(\mathbf{x}(0),\boldsymbol{\theta})] \\ \qquad -\dfrac{\Delta t}{2}\displaystyle\sum_{i=1}^{D}\frac{R_f(i)}{M+1}\cdot[x_i(1) - f_i(\mathbf{x}(0),\boldsymbol{\theta})]\cdot\mathcal{J}_{ia}(0,\boldsymbol{\theta}), \qquad \underline{m=0}\,; \\ \\ \dfrac{R_f(a)}{M+1}\cdot\Big\{[x_a(m) - f_a(\mathbf{x}(m-1),\boldsymbol{\theta})] - [x_a(m+1) - f_a(\mathbf{x}(m),\boldsymbol{\theta})]\Big\} \\ \qquad -\dfrac{\Delta t}{2}\displaystyle\sum_{i=1}^{D}\frac{R_f(i)}{M+1}\cdot\mathcal{J}_{ia}(m,\boldsymbol{\theta})\cdot\Big\{[x_i(m) - f_i(\mathbf{x}(m-1),\boldsymbol{\theta})] \\ \qquad +[x_i(m+1) - f_i(\mathbf{x}(m),\boldsymbol{\theta})]\Big\}, \qquad \underline{m=1,\ldots,M-1}\,; \\ \\ \dfrac{R_f(a)}{M+1}\cdot[x_a(M) - f_a(\mathbf{x}(M-1),\boldsymbol{\theta})] \\ \qquad -\dfrac{\Delta t}{2}\displaystyle\sum_{i=1}^{D}\frac{R_f(i)}{M+1}\cdot[x_i(M) - f_i(\mathbf{x}(M-1),\boldsymbol{\theta})] \\ \qquad \cdot\mathcal{J}_{ia}(M,\boldsymbol{\theta}), \qquad \underline{m=M}\,. \end{cases} \tag{2.16}$$

Note that in Eq. (2.16), we have generalized the original definition of R_f, allowing it to

take on different values for different a. This will be useful in numerical calculations.

For the derivatives with respect to $\theta_1, \ldots, \theta_{N_p}$, we have, for $b = 1, \ldots, N_p$,

$$\begin{aligned}\frac{\partial \,\mathrm{ModErr}(\mathbf{X})}{\partial \theta_b} = &-\sum_{i=1}^{D} \frac{R_f(i)}{M+1} \sum_{m=0}^{M-1} \left[x_i(m+1) - f_i(\mathbf{x}(m), \boldsymbol{\theta})\right] \\ &\times \frac{\Delta t}{2} \cdot \left[\mathcal{G}_{ib}(m, \boldsymbol{\theta}) + \mathcal{G}_{ib}(m+1, \boldsymbol{\theta})\right].\end{aligned} \tag{2.17}$$

2.2.3 The Goal Revisited

Let us now resume our previous discussion. Our challenge is to perform expected-value integrals such as Eq. (2.11). One should anticipate that the dominant contribution comes from the maxima of $\pi(\mathbf{X} \,|\, \mathbf{Y})$, or, equivalently, the minima of $A(\mathbf{X})$. Near such minima, the two contributions to the action, the measurement error and the model error, balance each other to accomplish the explicit transfer of information from the data to the model.

As in all practically interesting cases, when $\mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$ is nonlinear in $\mathbf{X}$, the expected-value integral Eq. (2.11) is non-Gaussian, and we need to approximately evaluate integrals of this form in order to complete the task of transferring information.

Two generally useful approaches for evaluating this kind of high dimensional integral are Laplace's method [42] and the collection of techniques using Monte Carlo sampling [33, 34, 35, 37, 43].

The Laplace-method evaluations of expected-value integrals is discussed in Refs. [44, 45, 46, 47, 48]. They do not sample from $\pi(\mathbf{X} \,|\, \mathbf{Y})$ away from its maximum. Evaluating corrections to the leading Laplace contributions is familiar as perturbation theory in statistical physics [49]. The convergence of such perturbation methods can depend sensitively on the landscape of the action in the $\mathbf{X}$ space. In addition, the need for repeatedly using the Hessian matrix of $A(\mathbf{X})$ in many methods may further hinder their use in practice.

In Sec. 2.3.1 and Sec. 2.3.2, we briefly review the Hamiltonian Monte Carlo method.

We then give a two-dimensional example comparing the original and the Hamiltonian Monte Carlo methods in Sec. 2.3.3. Then, in Sec. 2.4, we quantitatively formulate the proposed method to evaluate Eq. (2.11).

2.3 Hamiltonian Monte Carlo

Performing integrals such as Eq. (2.11) via Monte Carlo searches requires a method to sample from $\pi(\mathbf{X}\,|\,\mathbf{Y})$ in order to identify the regions in the path space $\mathbf{X}$ yielding the dominant contribution to the integral. In this section, we give an overview of the Hamiltonian Monte Carlo (HMC) method that is a functioning part of the proposed method of this dissertation. We also provide one basic example that illustrates the advantages of Hamiltonian Monte Carlo.

Before we begin, let us first briefly discuss why the original implementation of Markov chain Monte Carlo (MCMC), which we call the random-proposal (RP) Monte Carlo, is not a good choice.

The original Monte Carlo procedure devised by Metropolis and Hastings [34, 35, 50] has been extensively explored. It samples from a probability distribution such as $\pi(\mathbf{X}\,|\,\mathbf{Y})$ by proposing candidate states from the so-called proposal distributions. Interested readers could easily find relevant literature such as the papers and books cited here. In particular, proposals on how to move about path space are made by perturbing the present location at random. This random-proposal (RP) procedure is in theory ergodic, meaning that the proposals can reach any region in the path space with nonzero probability given a sufficient number of proposed steps [51]. The ergodicity makes random-proposal Monte Carlo an unbiased method for performing expected-value integrals. However, in practice, the RP procedure often suffers from slow mixing in high dimensional distributions [36, 52, 53].

As mentioned above, the random-proposal Monte Carlo proposes a new state from

a present path space position $\mathbf{X}_i$ to a new, candidate position $\mathbf{X}_c$ via a (usually symmetric) proposal distribution. Then, according to RP's acceptance rule assuring the detailed balance condition, accept $\mathbf{X}_c$ as the new sample $\mathbf{X}_{i+1}$ with probability

$$\alpha_{c\,|\,i} = \min\left\{1,\ \frac{\pi\left(\mathbf{X}_c\,|\,\mathbf{Y}\right)}{\pi\left(\mathbf{X}_i\,|\,\mathbf{Y}\right)}\right\}. \tag{2.18}$$

If the proposed $\mathbf{X}_c$ is rejected, then $\mathbf{X}_{i+1} = \mathbf{X}_i$; if accepted, then $\mathbf{X}_{i+1} = \mathbf{X}_c$. Regardless of the outcome, $\mathbf{X}_{i+1}$ serves as the starting point for a subsequent proposal procedure. All accepted paths are subsequently used in the evaluation of $\langle G(\mathbf{X})\rangle$.

The convergence of this procedure to the desired distribution $\pi(\mathbf{X}|\mathbf{Y})$ is often be slow. This is especially when $\mathbf{X}$ is high-dimensional. One reason is that when making proposal states, it treats the movements in the $\mathbf{X}$ space as random walks, often resulting in low acceptance rates and rather limited movements in the path space. This fact is tightly related to the phenomenon called "curse of dimensionality" in statistics and computer science [31, 32, 54, 55]. As a result, the asymptotic distribution $\pi(\mathbf{X}\,|\,\mathbf{Y})$ may, in practice, be unreachable.

As a side note, enforcing detailed balance, i.e., "reversibility," on a Monte Carlo proposal is a sufficient but not a necessary condition for convergence [56, 57, 58].

2.3.1 General Idea of HMC-Like Sampling

Hamiltonian Monte Carlo (HMC) [37, 38, 39] strikes out in a new direction. It adds to the path $\mathbf{X}$ an additional set of variables, which we call $\mathbf{P}$, and identifies a search in the $(\mathbf{X}, \mathbf{P})$ space that preserves some invariants as a result of some rules for moving about that space. This is explained quantitatively below. If, for example, the motion in the extended space preserved $\mathbf{P}\cdot\mathbf{P} + \mathbf{X}\cdot\mathbf{X}$, then moving about the space by performing rotations would allow us to move interesting distances in $(\mathbf{X}, \mathbf{P})$ space by simply rotating the variables in

the high dimensional extended space while staying on the $\mathbf{P}\cdot\mathbf{P}+\mathbf{X}\cdot\mathbf{X}=$ constant surface. Making this motion in the extended space may allow large moves in $\mathbf{X}$ space with large probabilities of acceptance. If this were the case, one might be sampling $\exp[-A(\mathbf{X})]$ much more efficiently than detailed balance-preserved random walking as in the random-proposal Monte Carlo procedures.

As introduced in Ref. [37], the added conjugate variables $\mathbf{P}$ are selected from the well explored examples of the *canonical momenta* familiar to physicists and thoroughly analyzed for two centuries. Note also that $\mathbf{P}$ has the same dimensions as $\mathbf{X}$, which is $D(M+1)+N_p$ as shown clearly in Eq. (2.4). By choosing the additional variables $\mathbf{P}$ to be canonical conjugates of $\mathbf{X}$, one can use the rules of classical mechanics to move around in $(\mathbf{X},\mathbf{P})$ space and preserve the underlying symplectic structure.

If the movement is labeled by a scalar s which we call "time," then the rule

$$\frac{d}{ds}\begin{pmatrix}\mathbf{X}(s)\\ \mathbf{P}(s)\end{pmatrix}=\begin{pmatrix}\mathbf{0} & \mathbf{I}\\ -\mathbf{I} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\nabla_{\mathbf{X}}H(\mathbf{X}(s),\mathbf{P}(s))\\ \nabla_{\mathbf{P}}H(\mathbf{X}(s),\mathbf{P}(s))\end{pmatrix} \tag{2.19}$$

preserves the value of the scalar function $H(\mathbf{X},\mathbf{P})$ as well as volumes in the $(\mathbf{X},\mathbf{P})$ space and a collection of other quantities known as the Poincaré invariants [59, 60]. It is worth noting that the antisymmetric matrix on the right-hand side of Eq. (2.19) can also be modified to construct new HMC-like proposals [61].

For HMC, in addition to the original target distribution $\pi(\mathbf{X}\,|\,\mathbf{Y})\propto\exp[-A(\mathbf{X})]$, one needs to choose an arbitrary distribution $\pi(\mathbf{P}\,|\,\mathbf{X},\mathbf{Y})$ for the canonical momenta $\mathbf{P}$ in order to fully specify the joint HMC target $\pi(\mathbf{X},\mathbf{P}\,|\,\mathbf{Y})$. The Hamiltonian can be written as

$$\begin{aligned}H(\mathbf{X},\mathbf{P}) &= -\log\left[\pi\left(\mathbf{X},\mathbf{P}\,|\,\mathbf{Y}\right)\right]\\ &= A(\mathbf{X})+h(\mathbf{P},\mathbf{X}),\end{aligned} \tag{2.20}$$

where $h(\mathbf{P},\mathbf{X})$ is $-\log[\pi(\mathbf{P}\,|\,\mathbf{X},\mathbf{Y})]$ up to an additive constant. Under the augmented

target distribution, i.e.,

$$\pi\left(\mathbf{X}, \mathbf{P} \,|\, \mathbf{Y}\right) \propto e^{-H(\mathbf{X},\mathbf{P})}, \tag{2.21}$$

the expected value of $G(\mathbf{X})$ is now

$$\begin{aligned} \left\langle G(\mathbf{X}) \right\rangle_{\pi(\mathbf{X},\,\mathbf{P}\,|\,\mathbf{Y})} &= \int d\mathbf{X}\, d\mathbf{P}\, G(\mathbf{X})\, \pi(\mathbf{P} \,|\, \mathbf{X}, \mathbf{Y})\, \pi(\mathbf{X} \,|\, \mathbf{Y}) \\ &= \left[\int d\mathbf{X}\, e^{-A(\mathbf{X})}\right]^{-1} \cdot \int d\mathbf{X}\, d\mathbf{P}\, \frac{G(\mathbf{X})\, e^{-H(\mathbf{X},\mathbf{P})}}{\int d\mathbf{P}\, e^{-h(\mathbf{P},\mathbf{X})}} \\ &= \left\langle G(\mathbf{X}) \right\rangle_{\pi(\mathbf{X}\,|\,\mathbf{Y})}, \end{aligned} \tag{2.22}$$

where $\left\langle G(\mathbf{X}) \right\rangle_{\pi(\mathbf{X}\,|\,\mathbf{Y})}$ has been defined in Eq. (2.11).

Eq. (2.22) tells us that the expected-value integrals, i.e., $\langle G(\mathbf{X}) \rangle$, are unchanged under HMC for an arbitrary choice of $h(\mathbf{P}, \mathbf{X})$. This tells us that once we have joint samples in the $(\mathbf{X}, \mathbf{P})$ space, we can easily recover the $\mathbf{X}$ samples that follow our original target distribution $\pi(\mathbf{X} \,|\, \mathbf{Y})$ by simply discarding the $\mathbf{P}$ parts.

The combination of symmetry-preserving movements in the $(\mathbf{X}, \mathbf{P})$ space and the invariance in the expected values is appealing. These properties make possible higher acceptance rates and faster mixing for sampling from $\pi(\mathbf{X} \,|\, \mathbf{Y})$. Other symmetries in $(\mathbf{X}, \mathbf{P})$ requiring different rules for generating motions might work equally well. HMC should be viewed as one of a class of approaches within this overall idea [62, 63]. It should also be noted that a high acceptance rate itself does not guarantee efficiency: combining high acceptance with fast exploration of phase space is a goal of all Markov chain Monte Carlo methods.

One must distinguish the use of the scalar label s in HMC from the time labels t and τ as described in Sec. 2.1: s is just a label to track movements in the enlarged space. Actual times t and τ are labels used in identifying the path of model dynamical variables $\mathbf{x}(t)$ and the data $\mathbf{y}(\tau)$ through an observation window.

2.3.2 HMC Itself

As discussed above, the HMC procedure [37, 38, 39] doubles the dimensionality of the path space, introducing physics-motivated but arbitrarily chosen canonical momenta $\mathbf{P}$ associated with path-space position $\mathbf{X}$.

While certainly not required, the choice for the "kinetic energy," $h(\mathbf{P}, \mathbf{X}) = \frac{\mathbf{P} \cdot \mathbf{P}}{2}$, is convenient and the resulting distribution $\exp[-h(\mathbf{P}, \mathbf{X})]$ is Gaussian. As studied in Ref. [64], small modifications of this standard choice to include higher order polynomials of $\mathbf{P}$ can introduce chaotic behavior and substantial mixing in motions generated by $H(\mathbf{X}, \mathbf{P})$ and thus may avoid some potential problems in the implementation of a quadratic choice for $h(\mathbf{P}, \mathbf{X})$. There are other interesting choices of $h(\mathbf{P}, \mathbf{X})$ that may improve the performance of HMC sampling [65, 66, 67].

We proceed with the quadratic choice of $h(\mathbf{P}, \mathbf{X})$. The Hamiltonian becomes

$$H(\mathbf{X}, \mathbf{P}) = A(\mathbf{X}) + \frac{\mathbf{P} \cdot \mathbf{P}}{2}. \tag{2.23}$$

Why does one want to work with the even larger $(\mathbf{X}, \mathbf{P})$ space given that $\mathbf{X}$ is already high-dimensional? The answer lies in the invariance properties of Hamiltonian dynamics. If HMC proposals are made using integration of Eq. (2.19) from "time" 0 to s with the choice of H as in Eq. (2.23), namely,

$$\frac{d}{ds} \begin{pmatrix} \mathbf{X}(s) \\ \mathbf{P}(s) \end{pmatrix} = \begin{pmatrix} \mathbf{P}(s) \\ -\nabla A(\mathbf{X}(s)) \end{pmatrix}, \tag{2.24}$$

then $H(\mathbf{X}, \mathbf{P})$ is conserved along the canonical phase-space orbit labeled by s. As discussed, many other quantities are preserved. Among them, HMC makes use of the conservation of phase-space volume, which means $d\,\mathbf{X}(s)\, d\,\mathbf{P}(s) = d\,\mathbf{X}(0)\, d\,\mathbf{P}(0)$.

The core of HMC is to propose $(\mathbf{X}(s), -\mathbf{P}(s))$ by forward-integrating the Hamiltonian

dynamics, starting from $(\mathbf{X}(0), \mathbf{P}(0))$. This process is precise when the integration is performed with s taken as a continuous variable, so a Hamiltonian flow is realized and $H(\mathbf{X}, \mathbf{P})$ is conserved. A complete HMC proposal includes a negative sign before $\mathbf{P}(s)$, meaning that we need to flip the momentum *after* the Hamiltonian integration is finished. This flipping is to make the proposal reversible in s and symmetric in $(\mathbf{X}, \mathbf{P})$, thus ensuring detailed balance [38].

In practice, the form of $A(\mathbf{X})$ precludes analytical evaluations, and one must work with a discrete s in order to integrate Eq. (2.24). A non-symplectic integrator is obviously not a choice because it does not produce any invariance. Still, when we use a symplectic integrator to perform the integration of Hamilton's equations in discrete s [62], a result of Ge and Marsden [68] tells us that we cannot precisely preserve both the symplectic form and $H(\mathbf{X}, \mathbf{P})$ at the same time. This result is intuitive—after all, only the exact Hamiltonian flow itself would preserve both the symplectic form and $H(\mathbf{X}, \mathbf{P})$. Consequently, one needs to decide in practice whether to preserve $H(\mathbf{X}, \mathbf{P})$ or to preserve the symplectic form.

The latter turns out to be a more sensible choice. The main reason is that if one chooses to preserve $H(\mathbf{X}, \mathbf{P})$ but gives up the symplectic form, then a non-Hamiltonian dynamics must be simulated, therefore it is hard to find a deterministic and time-reversible mapping needed to construct the detailed balance condition [62, 69, 70, 71].

Fig. 2.2 shows the results of using the second-order Runge-Kutta method [33] and the leapfrog method [38, 62], respectively, to integrate a simple Hamiltonian dynamics with $H(x, p) = \frac{x^2}{2} + \frac{p^2}{2}$. One can easily see that even with a much smaller step size, the Runge-Kutta method still results in a diverging trajectory and thus is not a good choice for HMC. On the contrary, the leapfrog method never strays far from the ground truth, preserving phase-space volume.

Overall, we expect that by using a symplectic integrator in discrete s, we will find $H(\mathbf{X}(s), \mathbf{P}(s)) \approx H(\mathbf{X}(0), \mathbf{P}(0))$, but not exactly equal, as shown clearly in Fig. 2.2(b). As

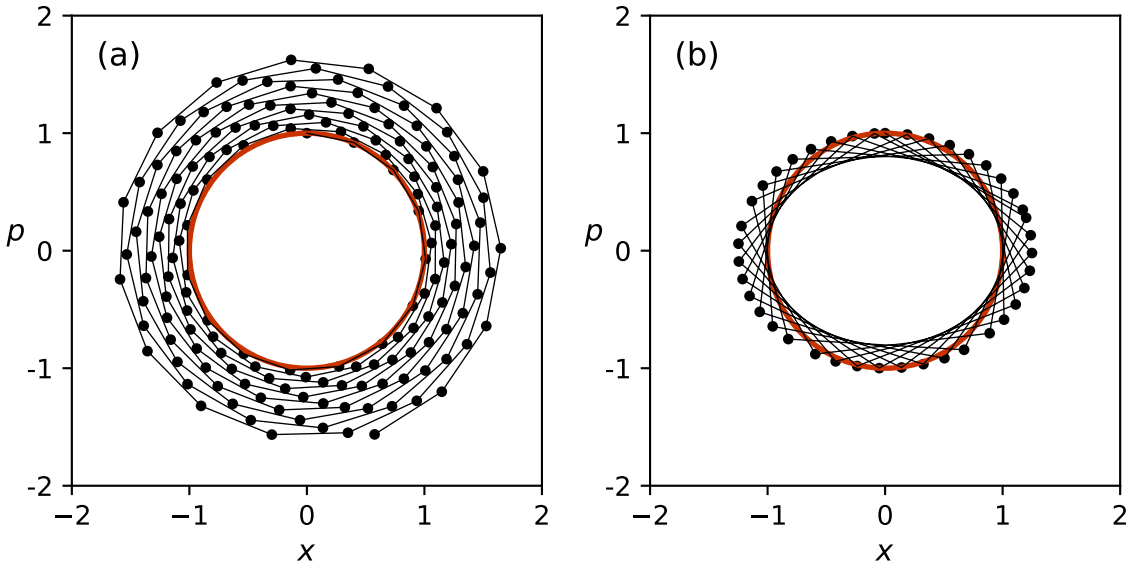

Figure 2.2: A two-dimensional example that compares two different methods for integrating Eq. (2.24). In both cases, the Hamiltonian is $H(x,p) = \frac{x^2}{2} + \frac{p^2}{2}$, as indicated by the red circles. (a) Using the second-order Runge-Kutta method [33]. The initial condition is $(0,1)$ and the step size is 0.4. There are 160 total steps shown. (b) Using the leapfrog method [38, 62]. The initial condition is $(0,1)$ and the step size is 1.2. There are 40 total steps shown.

a consequence, when determining the acceptance or rejection of the joint proposal state $(\mathbf{X}(s), \mathbf{P}(s))$, the acceptance probability can usually be near unity, but not precisely one, i.e.,

$$\begin{aligned} \alpha(\mathbf{X}(s), -\mathbf{P}(s) \,|\, \mathbf{X}(0), \mathbf{P}(0)) &= \min\left\{1, \frac{\exp\left[-H(\mathbf{X}(s), -\mathbf{P}(s))\right]}{\exp\left[-H(\mathbf{X}(0), \mathbf{P}(0))\right]}\right\} \\ &\approx 1. \end{aligned} \tag{2.25}$$

Comparing Eq. (2.25) with Eq. (2.18), we see that the overall acceptance rate of HMC is much higher than that of the random-proposal Monte Carlo method, especially when $\mathbf{X}$ is high-dimensional.

It is necessary to emphasize that as the parameters $\boldsymbol{\theta}$ are taken to be components of the path $\mathbf{X}$, they also have conjugate variables $\boldsymbol{\eta}$ included in $\mathbf{P}$. Therefore, HMC provides a principled manner of exploring $\pi(\mathbf{X} \,|\, \mathbf{Y})$ for both state variables and time-independent parameters.

2.3.3 HMC in Practice

We have just seen that when implementing HMC, one should select a symplectic integrator in order to preserve the symplectic invariants [38, 62, 63], and many choices are available. We discuss here a fairly common one, the leapfrog symplectic integrator [62, 72], which is known for its simplicity and accuracy. Under this choice, one selects a small step size ε in "time" s as the discretization interval.

Suppose that the most recent accepted path is $\mathbf{X}(0)$ and that $\mathbf{P}(0)$ is a new sample such that $\mathbf{P}(0) \sim \pi(\mathbf{P} \,|\, \mathbf{X}, \mathbf{Y})$, then $(\mathbf{X}(0), \mathbf{P}(0))$ is the starting point for proposing a new candidate proposal $(\mathbf{X}_c, \mathbf{P}_c)$ from the distribution $\pi(\mathbf{X}, \mathbf{P} \,|\, \mathbf{Y}) \propto \exp[-H(\mathbf{X}, \mathbf{P})]$.

The procedure to go from $(\mathbf{X}(0), \mathbf{P}(0))$ to $(\mathbf{X}_c, \mathbf{P}_c)$ is as follows. We first move from

$(\mathbf{X}(0), \mathbf{P}(0))$ to $(\mathbf{X}(\varepsilon), \mathbf{P}(\varepsilon))$ using the leapfrog symplectic stepping rule,

$$\begin{aligned} \mathbf{P}(\varepsilon/2) &= \mathbf{P}(0) - \frac{\varepsilon}{2} \nabla A(\mathbf{X}(0)), \\ \mathbf{X}(\varepsilon) &= \mathbf{X}(0) + \varepsilon\, \mathbf{P}(\varepsilon/2), \\ \mathbf{P}(\varepsilon) &= \mathbf{P}(\varepsilon/2) - \frac{\varepsilon}{2} \nabla A(\mathbf{X}(\varepsilon)). \end{aligned} \tag{2.26}$$

Then, starting from $(\mathbf{X}(\varepsilon), \mathbf{P}(\varepsilon))$, we move to the next point at $(\mathbf{X}(2\varepsilon), \mathbf{P}(2\varepsilon))$ using the same rule as in Eq. (2.26). After proceeding for S steps, we arrive at the joint state $(\mathbf{X}(S\varepsilon), \mathbf{P}(S\varepsilon))$. The candidate proposal,

$$(\mathbf{X}_c, \mathbf{P}_c) = (\mathbf{X}(S\varepsilon), -\mathbf{P}(S\varepsilon)), \tag{2.27}$$

is then accepted or rejected according to Eq. (2.25). The negative sign in front of $\mathbf{P}(S\varepsilon)$ in Eq. (2.27) has been explained in Sec. 2.3.2.

As a preview, when performing Precision Annealing Hamiltonian Monte Carlo (or PAHMC, which will be formally proposed in Sec. 2.4), the HMC procedure is repeated for each value of R_f in Eq. (2.13). In addition, we choose to perform N_I independent PAHMC calculations in parallel starting from N_I independent initializations.

Let us now look at a two-dimensional example that illustrates the advantages of HMC over the standard random-proposal Monte Carlo method. In our example, we applied both methods to sample from the distribution $\pi(x, y) \propto \exp[-(8x^2 + 6y^2 - 5)^2 - 3y^2)]$. We started both methods at $(x, y) = (0, 0.8)$. Each of these two methods then generates 500 samples, and the last 301 of these samples are retained and displayed in Fig. 2.3. We can see that the random-walk Monte Carlo method explores only a small percentage of the distribution while HMC explores most of the distribution. The advantages of HMC over random-walk Monte Carlo is clear from this elementary example.

The example in Fig. 2.3 also reminds us of an important fact: HMC typically achieves

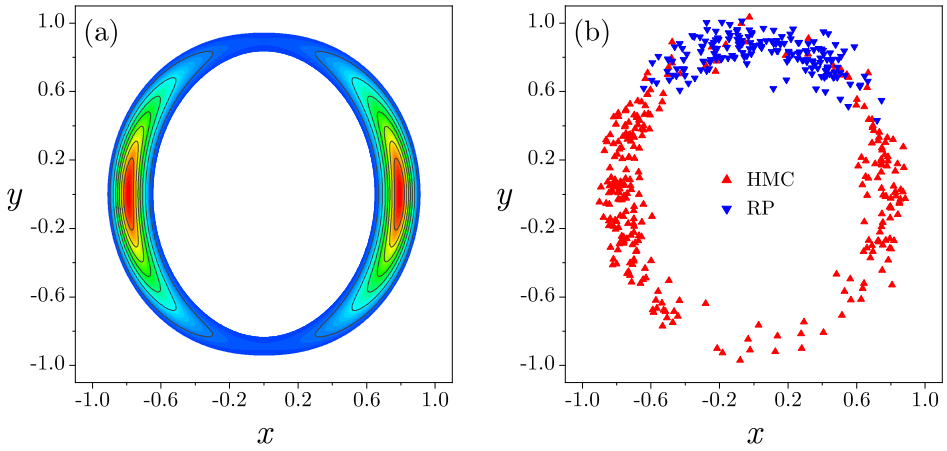

Figure 2.3: A comparison between Hamiltonian Monte Carlo and random-walk Monte Carlo. (a) A plot of the target distribution $\pi(x, y) \propto \exp[-(8x^2 + 6y^2 - 5)^2 - 3y^2)]$ within range $-1 \leq x, y \leq 1$. (b) The samples generated by random-proposal Monte Carlo are shown as blue, down-pointing triangles, and the samples generated by HMC are shown as red, up-pointing triangles. For HMC, to arrive at one proposal, the discrete Hamiltonian dynamics is simulated using the leapfrog method for $S = 50$ steps, with step size $\varepsilon = 0.01$. The overall acceptance rate for HMC is 0.998. For random-walk Monte Carlo, to arrive at one proposal, the x and y directions are perturbed simultaneously by drawing each perturbation from $\mathcal{N}(0, 0.01)$. The overall acceptance rate for random-walk Monte Carlo is 0.56.

better mixing than the random-proposal Monte Carlo method. Furthermore, it is important to compare the scaling properties of both methods as the dimensionality of the target distribution increases. With the potential applications to real-world problems in mind, it is critical that the sampling process we choose remains efficient in high dimensions. It has been shown that HMC scales better than random-walk Monte Carlo under many metrics. For example, after the equilibrium (of the Markov chain) is reached, for a d-dimensional target distribution, the computational complexity for a given level of acceptance rate typically grows as $d^{1.25}$ for HMC [73, 74, 75] and d^2 for RP [76]. This difference is one of the reasons why HMC is advantageous in many applications.

2.4 Precision Annealing

Although HMC is a powerful method, we will need much more to build a method that can achieve effective transfer of information when our negative log-likelihood is like Eq. (2.13).

In this section, we present the Precision Annealing (PA) framework, which is to be combined with HMC to achieve our goal of transferring information. We call the combined method Precision Annealing Hamiltonian Monte Carlo (PAHMC) [16]. We have discussed HMC in detail in Sec. 2.3, and we will discuss here the remaining part of the method, i.e., Precision Annealing. It is worth noting that both PA and HMC serve as functioning parts of the PAHMC method, but only when these two are combined together can we successfully transfer information from noisy data to a high-dimensional, nonlinear dynamical system that describes the underlying evolution of states.

The reason is straightforward. Because we only have a partial observation of the model state $\mathbf{x}(\tau_k)$ at each observation time τ_k, the unobserved degrees of freedom in the path $\mathbf{X}$ are completely uninformed upon initialization. If we impose $\mathbf{x}(m+1) = \mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$

to a high precision at the outset, the convoluted nonlinearities in the model may prevent any method from finding the desired global minimum in $A(\mathbf{X})$. Consequently, the dominant contribution to the integral in Eq. (2.11) cannot be identified and the information cannot be fully transferred.

It should be mentioned that ideas of a similar spirit have been proposed in earlier works [44, 48]. Nonetheless, in this dissertation, the Precision Annealing framework is proposed to serve two purposes as listed below.

(1) To guide the search for the global minimum of $A(\mathbf{X})$ by gradually recovering the "shape" (without changing the locations of the global and local minima!) of the $A(\mathbf{X})$ manifold in the path space.

(2) To refine the sampling behavior near the global minimum of $A(\mathbf{X})$ once the minimum is located. This is as important as (1) because the refinement is what drives down the model errors, as we will see in Chapter 3.

2.4.1 Initialization in Path Space

The strategy proposed in this dissertation is to vary the precision parameter R_f that enforces the model error term in $A(\mathbf{X})$, i.e., Eq. (2.13). When $R_f = 0$, the model is completely unresolved and the path space is highly degenerate for the unobserved degrees of freedom. In the opposite limit for the hyperparameter R_f where $R_f \to \infty$, the model $\mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$ becomes deterministic, and it enforces all nonlinear constraints prescribed by the model. Moving R_f from very small to quite large by small increments permits us to locate the maximum of $\pi(\mathbf{X} \,|\, \mathbf{Y})$ with high success rate and then perform HMC sampling in the vicinity.

Before we can start increasing R_f, we need to fully specify how $\mathbf{X}$ is initialized. We

now turn to this matter. At $R_f = 0$, Eq. (2.13) becomes a convex, quadratic form as

$$A(\mathbf{X}_{\text{init}}) = \sum_{k=0}^{F} \sum_{\ell=1}^{L} \frac{R_m}{2(F+1)} \left[x_\ell(k) - y_\ell(k)\right]^2 , \tag{2.28}$$

which apparently has its (degenerate) global minimum at $x_\ell(k) = y_\ell(k)$ for $k = 0, \ldots, F$ and $\ell = 1, \ldots, L$. Recall that a path $\mathbf{X}$ includes the set of D-dimensional state variables $\{\mathbf{x}(0), \ldots, \mathbf{x}(M)\}$ and the time-independent model parameters $\boldsymbol{\theta}$. Eq. (2.28) only specifies the initial values for L out of D dimensions at each time stamp and leaves the other components undetermined.

To fully specify our choice for the initial path, we now integrate the model forward for M discrete time increments. The first state $\mathbf{x}(0)$ is initialized such that $x_a(0) = y_a(0)$ for $a = 1, \ldots, L$; and for $a = L+1, \ldots, D$, $x_a(0)$ is drawn from a uniform distribution that covers the dynamical range of the model. The parameters $\boldsymbol{\theta}$ are also drawn from an appropriate uniform distribution. The model is then integrated forward in time, with the observed dimensions in $\mathbf{x}(m)$ replaced by y_ℓ at each time where we have a measurement, that is, when $t = \tau_k$ for $k = 0, \ldots, F$.

For the ease of notation, without loss of generality, we again let $n_k = k$ in Eq. (2.3) for $k = 0, \ldots, M$. Then, after $\mathbf{x}(0)$ is specified as above, we have, for $k = 0, \ldots, M-1$,

$$x_a(k+1) = \begin{cases} y_a(k+1), & \text{for } a = 1, \ldots, L, \\ f_a(\mathbf{x}(k), \boldsymbol{\theta}), & \text{for } a = L+1, \ldots, D. \end{cases} \tag{2.29}$$

This completes our construction of the initial path $\mathbf{X}_{\text{init}}$. The freedom in choosing $\mathbf{x}(0)$ and $\boldsymbol{\theta}$ at $R_f = 0$ gives us the flexibility to generate multiple such initial paths $\mathbf{X}_{\text{init}}^{(q)}$ for $q = 1, \ldots, N_I$. These N_I initial paths are retained, and they serve as the starting points of N_I independent calculations.

2.4.2 Procedure for Gradually Enforcing the Model

We have stated at the beginning of this section that Precision Annealing is a framework to (1) guide the search for the global minimum of $A(\mathbf{X})$ and (2) refine sampling in the vicinity of the global minimum. This subsection formally describes the procedure through which these two goals can be accomplished.

The core of the idea is as follows. We first begin with an $\mathbf{X}_{\text{init}}$ as described in Sec. 2.4.1. Then, we impose the model constraint $\mathbf{x}(m+1) = \mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$ slow enough by incrementally increasing R_f in Eq. (2.13). By doing so, there is a good chance that we can move to the "neighborhood" of the global minimum of $A(\mathbf{X})$ upon reaching certain R_f value. After that, as R_f continues to increase, the sampler keeps being at the vicinity of the global minimum of $A(\mathbf{X})$ and the model error term in Eq. (2.13) keeps decreasing. At the end of this procedure, the dominant contribution to Eq. (2.11) is identified with a high rate of success.

We now elaborate on the above by formulating the idea quantitatively. We adopt the following "schedule" for R_f:

$$R_f = R_{f_0}\, \alpha^{\beta} \tag{2.30}$$

with $\alpha > 1$ and $\beta = 0, 1, \ldots, \beta_{\max}$. Note that (1) α and $\beta_{\max}$ are hyper-parameters, and (2) R_{f_0} should be small. A choice of α near unity leads to the slow increase in R_f as β increases, which in turn brings in the nonlinearity of the model in an adiabatic manner. At each R_f value, N_I independent PAHMC calculations are to be performed, and each of these N_I calculations begins with the solution generated by the procedure under the previous R_f value.

Having discussed the "schedule" for R_f, we now take a close look at the first value at $\beta = 0$. First recall that at $R_f = 0$, for each of the N_I independent PAHMC calculations, we start from the corresponding initial path $\mathbf{X}_{\text{init}}^{(q)}$. Following that, at $R_f = R_{f_0}$, we sample

from our target distribution $\pi(\mathbf{X}\,|\,\mathbf{Y}) \propto \exp[-A(\mathbf{X})]$ using HMC, with $R_f = R_{f_0}$ in $A(\mathbf{X})$. We collect $N_{\{\beta=0\}}$ sampled paths, each denoted by $\mathbf{X}^{(q)}_{\{\beta=0\},\,j}$, with $j = 1, \ldots, N_{\{\beta=0\}}$ and $q = 1, \ldots, N_I$. These $N_{\{\beta=0\}}$ paths are generated by HMC. As well known in Monte Carlo calculations, we then take the sample mean of each of the N_I Markov chains developed so far, and we have

$$\overline{\mathbf{X}}^{(q)}_{\{\beta=0\}} = \frac{1}{N_{\{\beta=0\}}} \sum_{j=1}^{N_{\{\beta=0\}}} \mathbf{X}^{(q)}_{\{\beta=0\},\,j}\,, \tag{2.31}$$

for $q = 1, \ldots, N_I$.

Next, for each of the N_I independent calculations, we take $\overline{\mathbf{X}}^{(q)}_{\{\beta=0\}}$ as the initialization point at move to the next β, i.e., $\beta = 1$. This time, we collect $N_{\{\beta=1\}}$ sampled paths generated by HMC. Then, similarly, we have

$$\overline{\mathbf{X}}^{(q)}_{\{\beta=1\}} = \frac{1}{N_{\{\beta=1\}}} \sum_{j=1}^{N_{\{\beta=1\}}} \mathbf{X}^{(q)}_{\{\beta=1\},\,j}\,, \tag{2.32}$$

for $q = 1, \ldots, N_I$.

We continue this procedure until we reach the pre-determined $\beta_{\max}$ and have $\overline{\mathbf{X}}^{(q)}_{\{\beta=\beta_{\max}\}}$ ready for all $q = 1, \ldots, N_I$.

By plotting two quantities versus R_f or $\beta = \log_\alpha(R_f/R_{f_0})$, we will have insight on how to select $\beta_{\max}$. This is explained below.

First, we should make a plot of the action, as determined by each of the N_I paths at each R_f, as a function of β. We find that the action becomes independent of β (plateaus) as the precision of the model increases. The origin of this plateauing phenomenon can be seen in a second graph when plotting the model error term in the action, i.e.,

$$\text{ModErr}(\mathbf{X}) = \sum_{m=0}^{M-1} \sum_{a=1}^{D} \frac{R_f}{2M} \left[x_a(m+1) - f_a(\mathbf{x}(m), \boldsymbol{\theta})\right]^2, \tag{2.33}$$

as a function of β.

If the model faithfully describes the evolution of data in time, the model error will rapidly decrease to a small value as β increases. The action is the sum of the model error and the measurement error. As the model error tends numerically to a small number, the action levels out to the measurement error, which is independent of R_f. Indeed, the measurement error both provides a lower bound to the action and indicates whether the constructed model, $\mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$, is "correct." Note that this independence between the action and β can certainty change when we change L, i.e., the number of observed dimensions out of a total of D dimensions at each time.

As a result of these PAHMC calculations, the dominant contribution to the expected value of $G(\mathbf{X})$, as defined in Eq. (2.11), will be given by

$$\langle G(\mathbf{X}) \rangle \approx \frac{1}{N_I} \sum_{q=1}^{N_I} \overline{G(\mathbf{X})}^{(q)}_{\{\beta=\beta_{\max}\}}, \tag{2.34}$$

where, for $q = 1, \ldots, N_I$, we have

$$\overline{G(\mathbf{X})}^{(q)}_{\{\beta=\beta_{\max}\}} = \frac{1}{N_{\{\beta=\beta_{\max}\}}} \sum_{j=1}^{N_{\{\beta=\beta_{\max}\}}} G(\mathbf{X})^{(q)}_{\{\beta=\beta_{\max}\},j} \,. \tag{2.35}$$

The reason why Eq. (2.34) is true and useful is that PAHMC can identify the dominant probability mass of $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$ with a high rate of success. In Chapter 3, we will show numerical results that support this statement. These numerical calculations typically have $N_I \approx 30$.

Chapter 3

Numerical Results on Physical Systems

In Chapter 2, we discussed in great detail the proposed method for solving the problem of transferring information from noisy data into a physical dynamical system. Under the state-space representation, we derived a negative log-likelihood function $A(\mathbf{X})$, which we called the action, that contains all the information about the data and the dynamics. We then identified the goal to be evaluating the expected-value integral, i.e., Eq. (2.11), given the conditional distribution $\pi(\mathbf{X} \,|\, \mathbf{Y})$. The Precision Annealing Hamiltonian Monte Carlo (PAHMC) method itself was introduced in Sec. 2.3 and Sec. 2.4.

Our focus in this chapter is the applications of the PAHMC method. In particular, we present two sets of numerical result that collectively demonstrate the efficacy of the PAHMC method. The first set of result is related to a partially observed chaotic dynamical system while the second is related to a sparsely observed biophysical dynamical model of a neuron cell.

In the results to be discussed, and in almost all real-world information transfer problems, the challenges arise from the following facts.

(1) The data $\mathbf{Y}$ are noisy and incomplete.

(2) The path $\mathbf{X}$ is very high-dimensional.

(3) The action $A(\mathbf{X})$ is non-convex due to the nonlinearities in the dynamics $\mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$.

The first on the list has been mentioned at the beginning of Sec. 2.2. The rest are inherent computational difficulties that we need to overcome. These challenges make the results in this chapter meaningful as they are good test beds for the PAHMC method.

Before we begin, it is worth stating again what quantities are to be estimated along the evaluation of the expected-value integral. As we have seen in Sec. 2.1, all of the components within the path $\mathbf{X}$ need to be estimated. These include all the *observed* and *unobserved* state variables at all discrete time stamps within $t_0 \leq t \leq t_{\text{final}}$ as well as all the time-independent *parameters* in $\boldsymbol{\theta}$.

3.1 The Lorenz96 Chaotic Dynamical System

We present in this section a series of numerical results on transferring information from simulated data into a chaotic dynamical system using PAHMC. The dynamical system is commonly referred to as Lorenz96 [77, 78], and it is widely discussed in the geophysical literature and is an ideal test bed for vetting new statistical physics approaches. As we will see in a moment, the dynamical equations of Lorenz96 are simple and elegant, yet the chaotic nature makes it hard to achieve good estimation and prediction results.

Edward N. Lorenz proposed and subsequently studied a collection of ordinary differential equations in 1996 [77], and these famous ODEs are now known as Lorenz96. The state variables of Lorenz96 are set out on a ring, presumably representing equatorial locations [1]. This ODE system is meant to mimic the forced westward flow in the state variables transported by a quadratic "advection." Lorenz and Emanuel [78] subsequently used it to discuss strategies in selecting additional locations for dynamical weather measurements

that are the most meaningful.

The Lorenz96 system has a D-dimensional state variable $\mathbf{x}(t) = (x_1(t), \ldots, x_D(t))$ on a ring and a fixed forcing parameter ν. The dynamical equations are

$$\frac{dx_a(t)}{dt} = x_{a-1}(t)\left(x_{a+1}(t) - x_{a-2}(t)\right) - x_a(t) + \nu, \qquad a = 1, \ldots, D, \tag{3.1}$$

where $x_{-1}(t) \equiv x_{D-1}(t)$, $x_0(t) \equiv x_D(t)$, and $x_1(t) \equiv x_{D+1}(t)$.

An obvious solution to Eq. (3.1) is $x_a(t) = \nu$ for $a = 1, \ldots, D$, which is a fixed-point solution. If we add a small perturbation, i.e., $x_a(t) = \nu + \delta x_a(t)$, Eq. (3.1) yields

$$\frac{d\,\delta x_a(t)}{dt} = \nu \cdot (\delta x_{a+1}(t) - \delta x_{a-2}(t)) - \delta x_a(t) + \text{second-order terms}, \tag{3.2}$$

for $a = 1, \ldots, D$. If we subsequently write

$$\delta x_a(t) = A_k(t)\, e^{ika}, \tag{3.3}$$

then the amplitude $A_k(t)$ satisfy

$$\frac{dA_k(t)}{dt} = A_k(t)\left[\nu \cdot \left(e^{ik} - e^{-2ik}\right) - 1\right]. \tag{3.4}$$

By examining the real part of Eq. (3.4), we know that the fixed-point solution $x_a(t) = \nu$ becomes unstable when $\nu > 8/9$. Furthermore, chaotic solution begins to appear when $\nu \approx 6.5$.

Fig. 3.1 shows a three-dimensional projection of a $D = 20$ Lorenz96 time series with a forcing parameter $\nu = 8.17$, generated by integrating Eq. (3.1) with a discretization interval of $\Delta t = 0.025$. From the figure, we can see that the system is chaotic, but it is localized and has structure in the phase space.

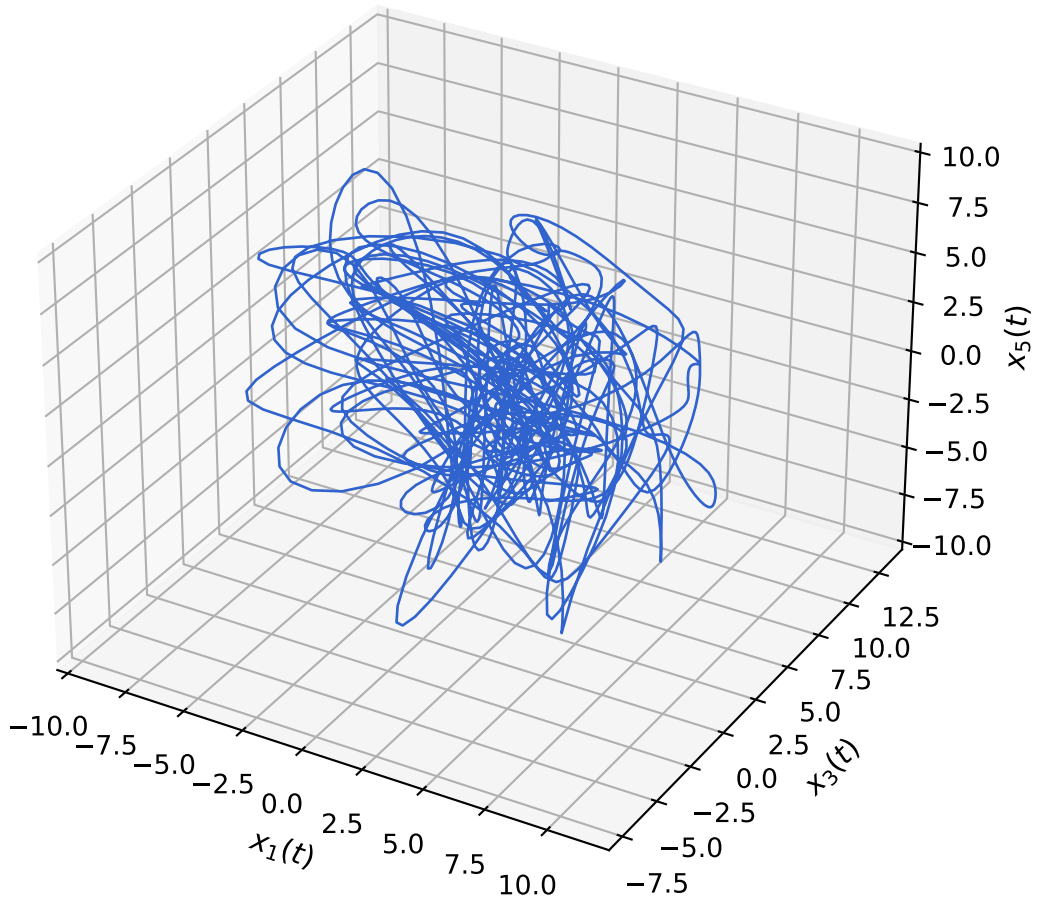

Figure 3.1: A three-dimensional projection of a $D = 20$ Lorenz96 time series with a forcing parameter $\nu = 8.17$. The integration interval is $\Delta t = 0.025$, and the length of the time series presented is 1000. Among the 20 dimensions, $x_1(t)$, $x_3(t)$, and $x_5(t)$ are shown.

3.1.1 Data Preparation

The numerical results that we will see in Sec. 3.1.2 through Sec. 3.1.6 are based on simulated data of Lorenz96. When performing the transfer of information, the PAHMC method is presented with a partially observed, noisy time series (whose collection is called $\mathbf{Y}$ as seen in Chapter 2). The true forcing parameter ν_{true} is not given to the method either and has to be "guessed" by the method. This way of validating a statistical physics approach is conveniently called a "twin experiment" in the literature [1].

We now turn to the description of the Lorenz96 data set. We choose $D = 20$, $\nu_{\text{true}} = 8.17$, and $\Delta t = 0.025$ for integrating Eq. (3.1). The observation window runs from $t_0 = 0$ to $t_{\text{final}} = 5$. The prediction window runs from $5 \leq t \leq 11$.

To each of the D time series generated, we add Gaussian noise with standard deviation $\sigma = 0.4$ throughout the observation and prediction windows.

We select $L < D$ components in the simulated Lorenz96 time series as the observations within the observation window. This L-time series will be our data, i.e., $\mathbf{Y}$. For each time within the observation window, we seek to estimate all D state variables. We also need to estimate the forcing parameter ν.

Assuming at each model time step $t = t_k$ we have an observation $\mathbf{y}(k)$, i.e., $n_k = k$ in Eq. (2.3) for $k = 1, \ldots, M$, then, the corresponding action is

$$\begin{aligned} A(\mathbf{X}) = &\sum_{k=0}^{F} \sum_{\ell=1}^{L} \frac{R_m}{2(F+1)} \left[x_\ell(\tau_k) - y_\ell(\tau_k) \right]^2 \\ &+ \sum_{m=0}^{M-1} \sum_{a=1}^{D} \frac{R_f}{2M} \left[x_a(m+1) - f_a(\mathbf{x}(m), \nu) \right]^2 . \end{aligned} \tag{3.5}$$

As usual, the first term in Eq. (3.5) represents the measurement error, and the second, the model error.

The trapezoidal rule is used to discretize Eq. (3.1), which means

$$f_a(\mathbf{x}(m),\nu) = x_a(m) + \frac{\Delta t}{2}\left[\mathcal{F}_a(\mathbf{x}(m+1),\nu) + \mathcal{F}_a(\mathbf{x}(m),\nu)\right], \tag{3.6}$$

for $a = 1, \ldots, D$, where $\mathcal{F}_a(\mathbf{x},\nu)$ is the model vector field.

It is worth restating that we do not simply fit a single parameter ν in the series of numerical examples. Instead, for each case, we need to estimate *all* the $D(M+1)+1$ degrees of freedom in the path $\mathbf{X}$. A detailed discussion on why this is necessary will be given in Chapter 4. As a matter of fact, the present problem is very high-dimensional albeit having only one parameter, ν.

We will present results for $L = 7$, 8, 10, and 12, respectively.

3.1.2 Results from PAHMC; $L = 7$

We first begin with $L = 7$ observed noisy time series out of $D = 20$. Specifically, we used the seven data series $y_\ell(m)$ with $\ell \in \{1, 4, 7, 10, 13, 16, 19\}$ as our observations. In Fig. 3.2(a), we display the action levels as a function of R_f for $L = 7$. In this figure, many action levels are present, which correspond to many peaks in the probability distribution $\pi(\mathbf{X}\,|\,\mathbf{Y}) \propto \exp[-A(\mathbf{X})]$ being identified. At large R_f values, the $N_I = 30$ action levels start to plateau. However, they are all well above the expected action value, which is equal to the measurement error component of the action. This is easy to prove given the Gaussian nature of the added noise and properties of the χ^2 distribution.

This result is an indication that although PAHMC has located some paths that agree with the Lorenz96 system, it fails to transfer information from the observations $\mathbf{Y}$ to the system due to an insufficient number of observations at each observation time τ_k. We expect that as the number of observations L increases, more information will become available to the PAHMC method, and the behavior of the action levels will change accordingly.

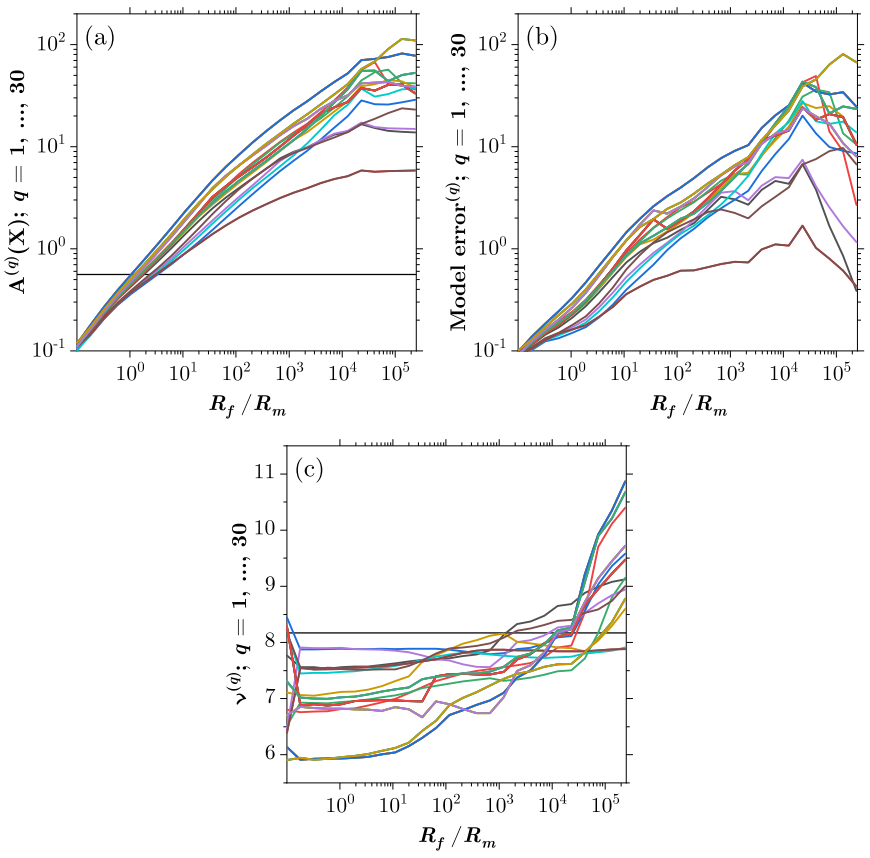

Figure 3.2: Action, measurement error, and model error for PAHMC on the Lorenz96 model with $D = 20$ and $L = 7$. Each step in the Precision Annealing procedure is associated with one value of R_f. We perform $N_I = 30$ independent calculations starting at N_I different $(\mathbf{X}(0), \mathbf{P}(0))$. (a) Action levels versus R_f. Because N_I calculations are performed at each R_f, many action curves are displayed. All the action levels are above the expected value (solid black line), which is the value of the measurement error term of the total action. (b) The model errors in Eq. (3.5) as a function of R_f. The model errors for the seven observed state variables remain large for all R_f. (c) The estimations of the Lorenz96 model forcing parameter ν as a function of R_f. The true forcing parameter is $\nu_{\text{true}} = 8.17$ (solid black line). At $L = 7$, the estimates of the forcing parameter are not accurate.

The inadequacy of using only 7 observations at each measurement time is also shown in the model error failing to significantly decrease as R_f is increasing. This is seen in Fig. 3.2(b). Similarly, in the estimation of the model forcing parameter shown in Fig. 3.2(c), we see substantial inaccuracies.

Furthermore, in the prediction results for an observed state variable $x_7(t)$, shown in Fig. 3.3(a), and for an unobserved state variable $x_{14}(t)$, shown in Fig. 3.3(b), we see many errors in both the estimation window $0 \leq t \leq 5$ and the prediction window $5 < t \leq 11$.

We will see next that as the number of observed dimensions increases, the overall results as well as the predictability also increases. This should be expected because more observations will become available as L increases.

3.1.3 Results from PAHMC; $L = 8$

In Fig. 3.4(a), we present the $L = 8$ action levels. Here, 7 out of 30 action levels split from the rest at an early stage in the Precision Annealing procedure, and these seven levels coincide with the anticipated action level for the measurement error term. For these agreed action levels, it implies that the transfer of information has been successful. Indeed, as we will see in the figures that follow, the estimations in the model parameter ν as well as the predictions beyond the observation window show that the paths with lowest action levels yield consistent results. In addition, Fig. 3.4(b) shows the model error calculations for $L = 8$. It is clear from there that the 7 lowest action levels are numerically dominated by their respective measurement errors instead of by model errors.

Fig. 3.4(c) shows the N_I estimations of the Lorenz96 forcing parameter, ν. The true value is $\nu_{\text{true}} = 8.17$. If we compare this plot with Fig. 3.4(a), we can conclude that although the PAHMC method has located some paths with high model precision (low model error), some of the paths (23 out of 30) still differ from the observations. The "wrong" models have been identified primarily because there are not enough measurements,

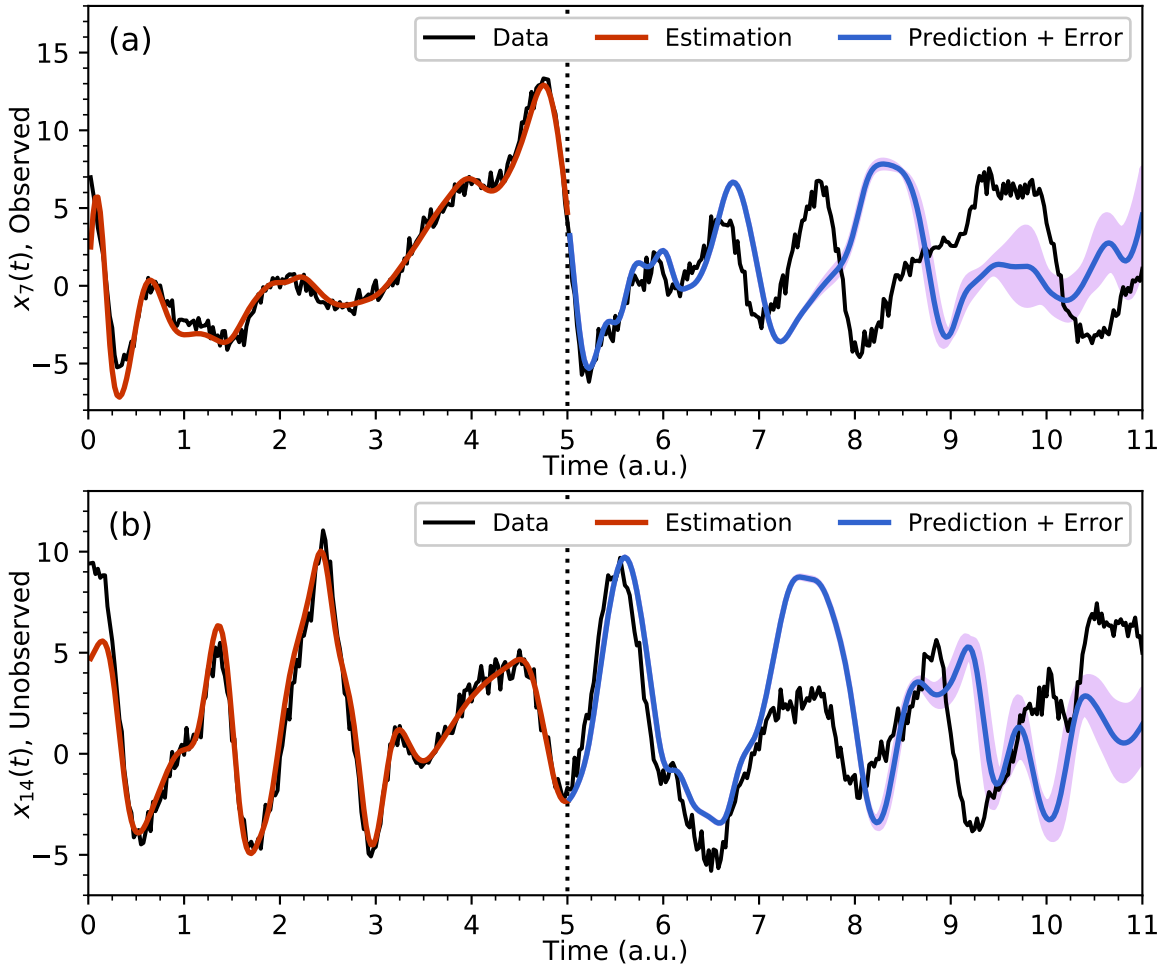

Figure 3.3: Predictions by PAHMC on the Lorenz96 model with $D = 20$ and $L = 7$. (a) Display of results for an observed state variable, $x_7(t)$. The noisy data are within in the time interval $0 \leq t \leq 11$. The estimated values in the observation window $0 \leq t < 5$ are shown in red. The predicted values are shown in blue, and the prediction errors are shown in magenta within $5 < t \leq 11$. (b) Display of results for an unobserved state variable, $x_{14}(t)$. The true data, with noise added, span the full $0 \leq t \leq 11$ interval. The estimated values in the observation window $0 \leq t < 5$ are shown in red. The predicted values, with errors (in magenta), are shown in blue within the interval of $5 < t \leq 11$. For the unobserved variables, the data within $0 \leq t \leq 11$ are unavailable to the method.

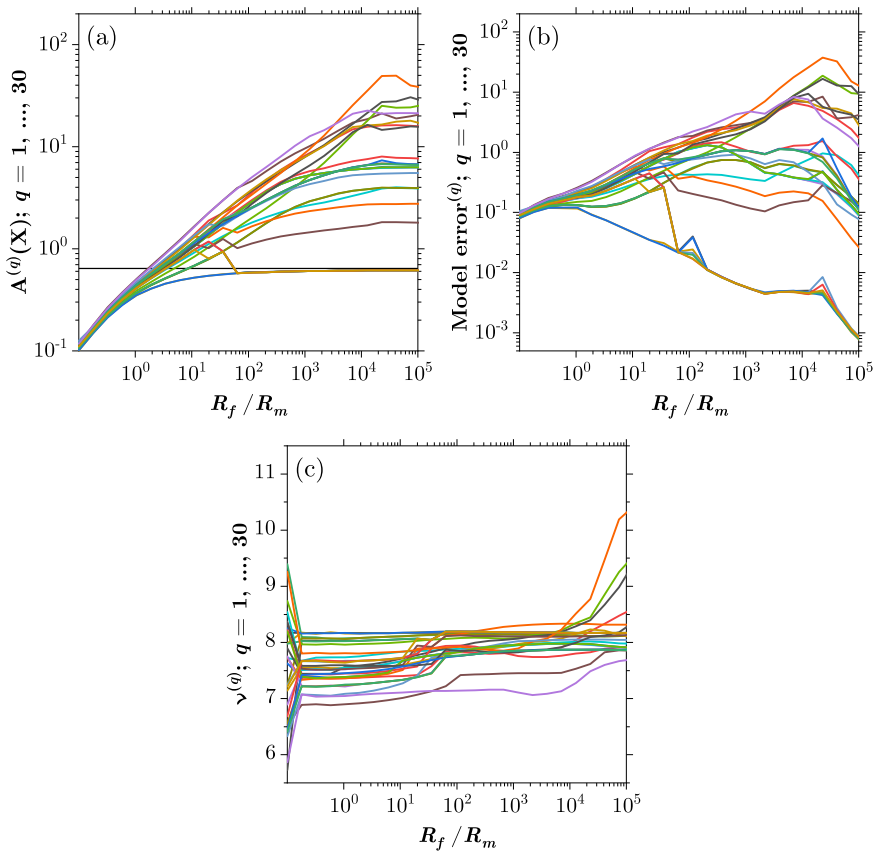

Figure 3.4: Action, measurement error, and model error for PAHMC on the Lorenz96 model with $D = 20$ and $L = 8$. Each step in the Precision Annealing procedure is associated with one value of R_f. We perform $N_I = 30$ independent calculations starting at N_I different $(\mathbf{X}(0), \mathbf{P}(0))$. (a) Action levels versus R_f. 23 of 30 action levels are above the anticipated value, shown as the solid black line. (b) The model errors in Eq. (3.5) as a function of R_f. At smaller R_f values, the model error dominates the action, indicating that the HMC calculations have not yet found a path that agrees with the model described by Eq. (3.6). At the final stage, the model error starts to decrease exponentially and the paths proposed by HMC start to agree with the model. (c) The estimations of the Lorenz96 model forcing parameter ν as a function of R_f. The true forcing parameter is $\nu_{\text{true}} = 8.17$ (solid black line).

in terms of L, at each time an observation is made. This leads to errors in the estimated forcing parameter.

Until this point, we have examined the indicators, i.e., action level, model error, and forcing parameter, representing the quality of the transfer of information. All of these are inferred from the calculations happening in the observation window $0 \leq t \leq 5$ and from the data $\mathbf{Y}$ available to the PAHMC method. To validate our information transfer method, we need to take advantage of the "twin experiment" setting introduced in Sec. 3.1.1. We will compare the estimation results for the observed and unobserved state variables with the ground-truth data generated in the twin experiment. We will also predict forward in time beyond the observation window (based on the estimation results) and compare the predictions with the ground truth.

Fig. 3.5(a) shows the estimate of an observed variable $x_9(t)$ within the estimation window $0 \leq t \leq 5$ as well as its prediction within the window $5 < t \leq 11$, for Lorenz96 with $L = 8$. We can see that the estimation agrees quite well with the data. The prediction, with the HMC calculation of errors shown, agrees with the data in the prediction window for some time (about 4 inverse Lyapunov times) and eventually diverges due to the chaotic nature of the Lorenz96 system at $\nu = 8.17$. It is worth noticing that the errors in the prediction are quite small in earlier stages. This is because the PAHMC procedure has accurately estimated all of the observed and unobserved state variables as well as the forcing parameter, ν.

Fig. 3.5(b) presents the same information, but for an *unobserved* variable $x_{12}(t)$. In this case, the data for $x_{12}(t)$ is not presented to the PAHMC method in the observation window, yet PAHMC still achieves high accuracy in terms of both estimation and prediction. This is a strong indication that the transfer of information from the data, $\mathbf{Y}$, to the Lorenz96 system has been successful for $L = 8$ observations. The nonlinearity of the model is at work here transferring information from observed to unobserved state variables.

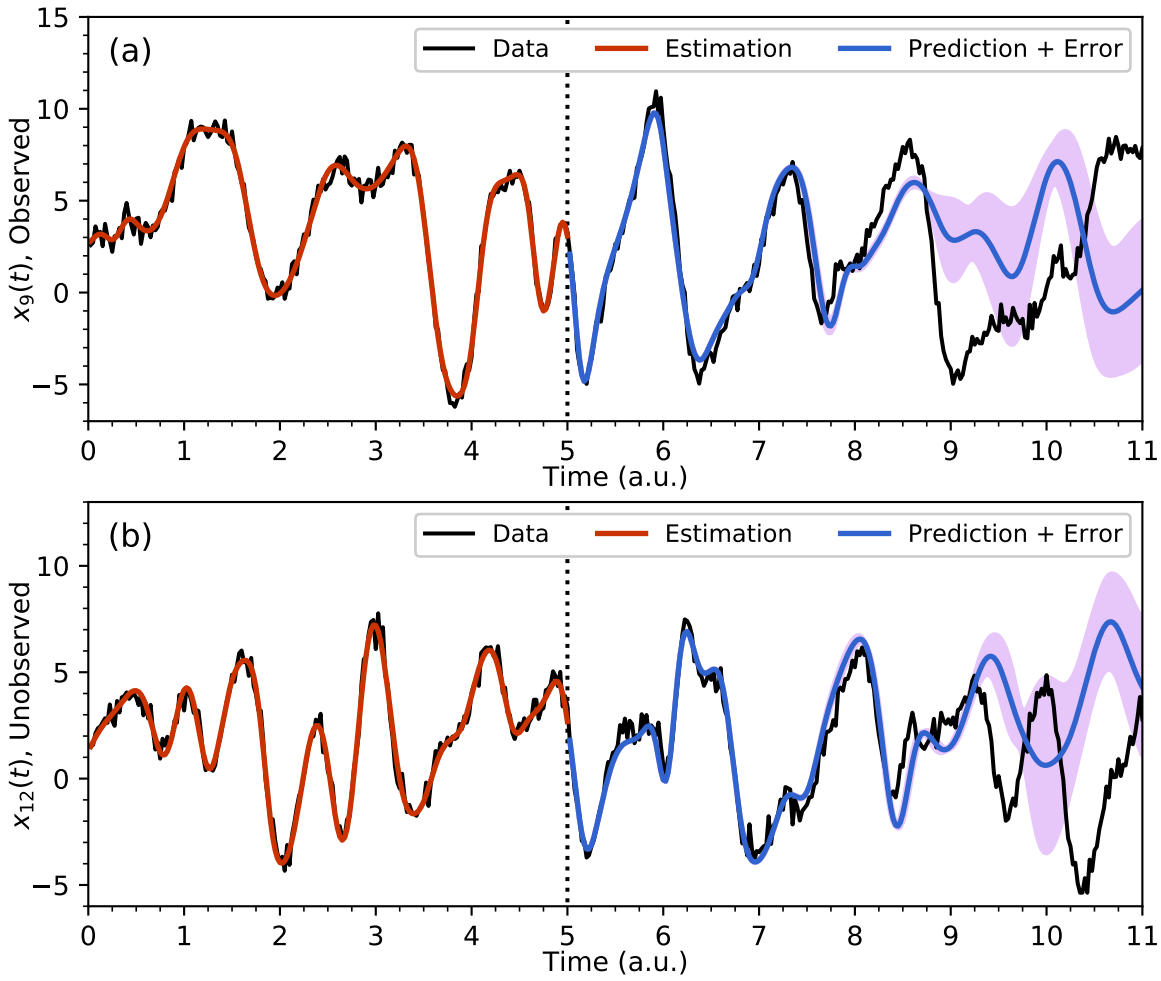

Figure 3.5: Predictions by PAHMC on the Lorenz96 model with $D = 20$ and $L = 8$. (a) Display of results for an observed state variable, $x_9(t)$. The noisy data are within in the time interval $0 \leq t \leq 11$. The estimated values in the observation window $0 \leq t < 5$ are shown in red. The predicted values are shown in blue, and the prediction errors are shown in magenta within $5 < t \leq 11$. (b) Display of results for an unobserved state variable, $x_{12}(t)$. The true data, with noise added, span the full $0 \leq t \leq 11$ interval. The estimated values in the observation window $0 \leq t < 5$ are shown in red. The predicted values, with errors (in magenta), are shown in blue within the interval of $5 < t \leq 11$. For the unobserved variables, the data within $0 \leq t \leq 11$ are unavailable to the method.

3.1.4 Results from PAHMC; $L = 10$

If we further increase the number of observations to $L = 10$, we shall see that all the action levels plateau almost at the expected value, as shown in Fig. 3.6(a). This indicates that the information is sufficient for the PAHMC method to locate the global minimum of the action. Also, in Fig. 3.6(b), the rapidly decaying model errors show that the dynamics, $\mathbf{x}(m+1) = \mathbf{f}(\mathbf{x}(m), \nu)$, is enforced strictly in the large-R_f regime. Fig. 3.6(c) shows the estimated forcing parameter for $L = 10$ as a function of R_f. The 30 estimated forcing parameters quickly converge to the true value of 8.17, which indicates that the correct model has been found by all the $N_I = 30$ PAHMC calculations. Conclusions similar to the $L = 8$ case can be drawn from the prediction results in Figs. 3.7(a) for the observed variable $x_{17}(t)$ and 3.7(b) for the unobserved state variable $x_{14}(t)$.

3.1.5 Results from PAHMC; $L = 12$

We have seen that $L = 10$ is sufficient for the number of observed dimensions for the $D = 20$ Lorenz96 system. As a result, increasing L to 12 brings almost no additional information about the properties of the source of the data or the structure of the action.

In Fig. 3.8 we can see the results for the action, model errors, and the forcing parameter for the $N_I = 30$ initial paths used in the calculations. Similar to the $L = 10$ case in Fig. 3.6, all 30 action levels rapidly converge to the expected value, and all model errors quickly become negligible. Although initialized at different values, the estimations for the forcing parameter converge to the true value 8.17 in the first few β values.

Fig. 3.9 shows the data, estimation and prediction results for one observed dimension, $x_{14}(t)$, and one unobserved dimension, $x_{11}(t)$. Within both the observation window and the prediction window, the results are good (note that for $x_{11}(t)$ there was no data). It is clear that the information transfer is successful.

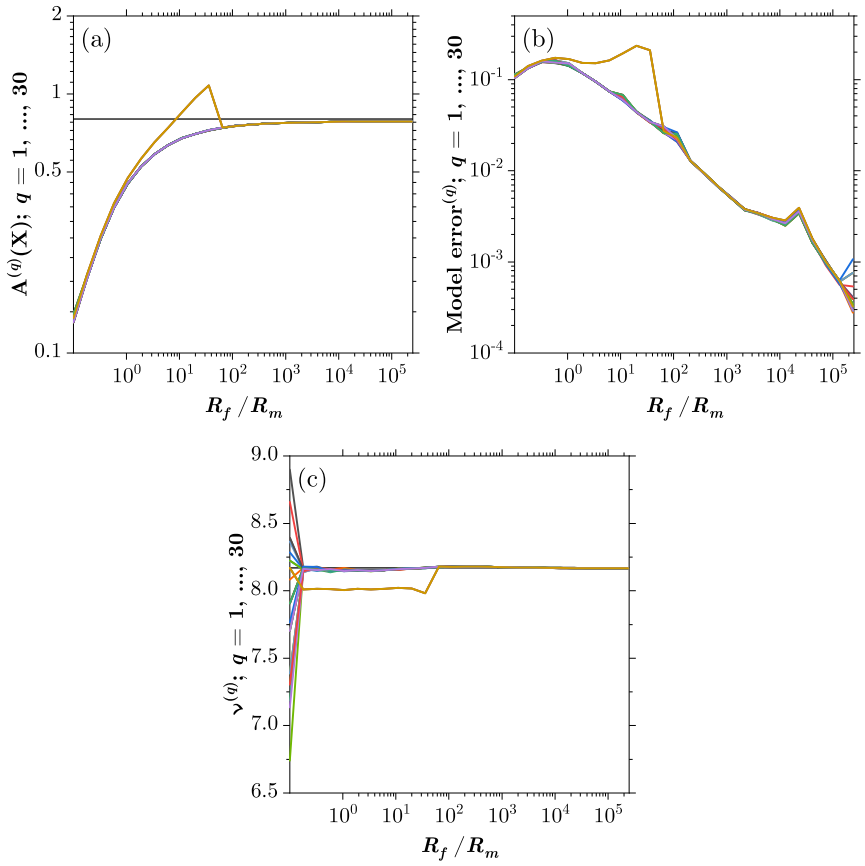

Figure 3.6: Action, measurement error, and model error for PAHMC on the Lorenz96 model with $D = 20$ and $L = 10$. Each step in the Precision Annealing procedure is associated with one value of R_f. We perform $N_I = 30$ independent calculations starting at N_I different $(\mathbf{X}(0), \mathbf{P}(0))$. (a) Action levels versus R_f. All these levels agree with the expected value, shown as the solid black line. (b) The model errors in Eq. (3.5) as a function of R_f. At smaller R_f values, the model error dominates the action. After that, the model error rapidly decreases as R_f grows, and the measurement error term dominates the action and becomes essentially independent of R_f. (c) The estimations of the Lorenz96 model forcing parameter ν as a function of R_f. The true forcing parameter is $\nu_{\text{true}} = 8.17$ (solid black line). The correct forcing parameter quickly emerges from the random initialization.

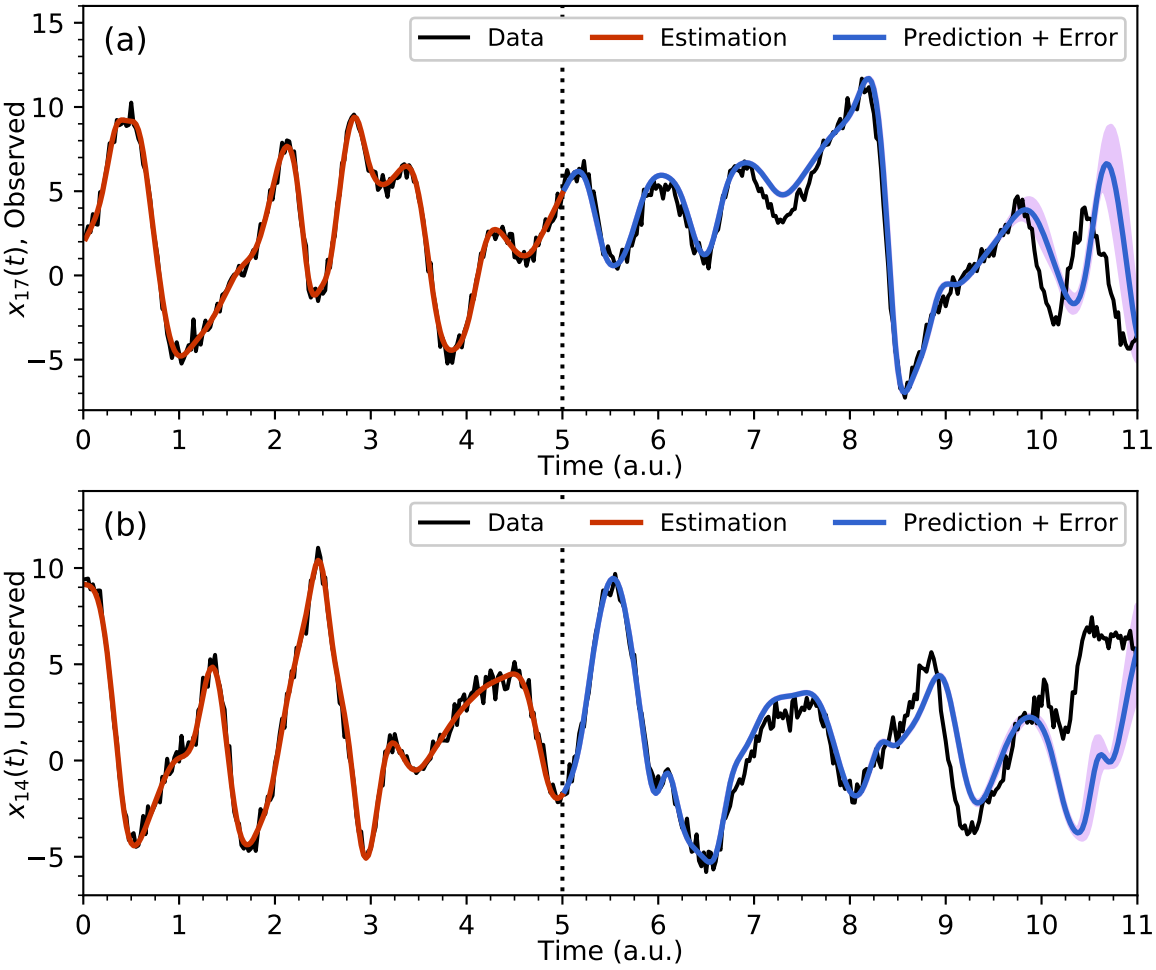

Figure 3.7: Predictions by PAHMC on the Lorenz96 model with $D = 20$ and $L = 10$. (a) Display of results for an observed state variable, $x_17(t)$. The data, with noise added, span the full $0 \leq t \leq 11$ interval. The estimated values in the observation window $0 \leq t < 5$ are shown in red. The predicted values, with errors (in magenta), are shown in blue within the interval of $5 < t \leq 11$. (b) Display of results for an unobserved state variable, $x_{14}(t)$. The true data, with noise added, span the full $0 \leq t \leq 11$ interval. The estimated values in the observation window $0 \leq t < 5$ are shown in red. The predicted values, with errors (in magenta), are shown in blue within the interval of $5 < t \leq 11$. For the unobserved variables, the data within $0 \leq t \leq 11$ are unavailable to the method.

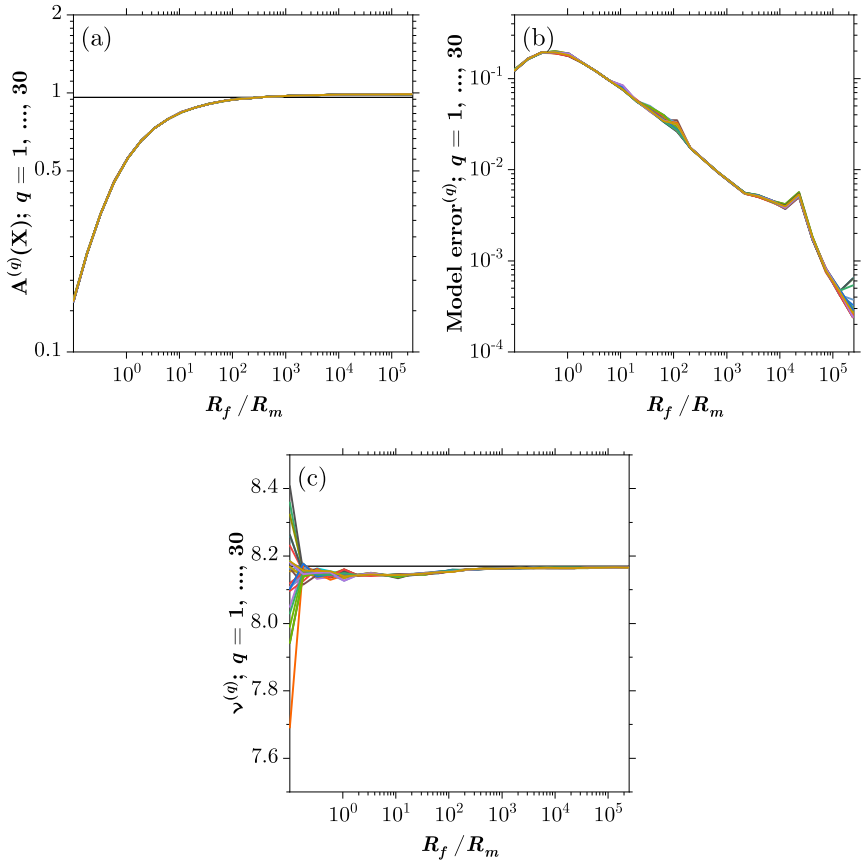

Figure 3.8: Action, measurement error, and model error for PAHMC on the Lorenz96 model with $D = 20$ and $L = 12$. Each step in the Precision Annealing procedure is associated with one value of R_f. We perform $N_I = 30$ independent calculations starting at N_I different $(\mathbf{X}(0), \mathbf{P}(0))$. (a) Action values at each R_f. All the action levels quickly plateau at the anticipated level as R_f grows. (b) The model errors in Eq. (3.5) as a function of R_f. At smaller R_f values, the model error dominates the action. After that, the model error rapidly decreases as R_f grows, and the measurement error term dominates the action and becomes essentially independent of R_f. This indicates that the Precision Annealing procedure has successfully located the path that agrees well with the measurements and the model. (c) The estimations of the Lorenz96 model forcing parameter ν as a function of R_f. The true forcing parameter is $\nu_{\text{true}} = 8.17$ (solid black line). The correct forcing parameter quickly emerges from the random initialization.

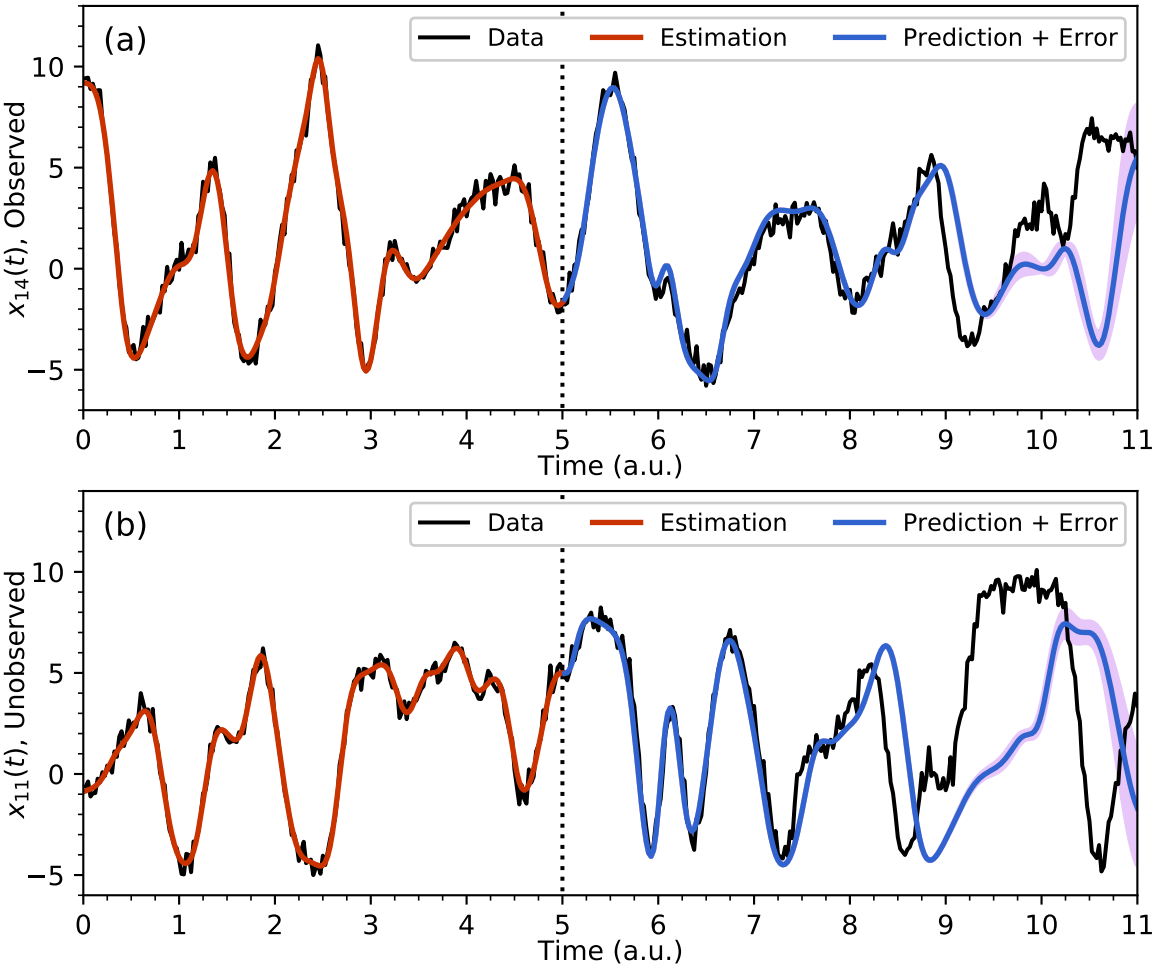

Figure 3.9: Predictions by PAHMC on the Lorenz96 model with $D = 20$ and $L = 12$. (a) Display of results for an observed state variable, $x_14(t)$. The data, with noise added, span the full $0 \leq t \leq 11$ interval. The estimated values in the observation window $0 \leq t < 5$ are shown in red. The predicted values, with errors (in magenta), are shown in blue within the interval of $5 < t \leq 11$. (b) Display of results for an unobserved state variable, $x_{11}(t)$. The true data, with noise added, span the full $0 \leq t \leq 11$ interval. The estimated values in the observation window $0 \leq t < 5$ are shown in red. The predicted values, with errors (in magenta), are shown in blue within the interval of $5 < t \leq 11$. For the unobserved variables, the data within $0 \leq t \leq 11$ are unavailable to the method.

3.1.6 Random-Proposal Monte Carlo Compared to PAHMC

Although the focus of this chapter is the applications of the Precision Annealing Hamiltonian Monte Carlo method, it is still useful to list some results from the Precision Annealing Random-proposal Monte Carlo method as a comparison. We have mentioned at the beginning of Sec. 2.3 that the Random-proposal Monte Carlo method (1) converges to the target distribution $\pi(\mathbf{X}\,|\,\mathbf{Y})$ slower than the HMC method and (2) scales poorly when the dimensionality of the problem grows. The examples in this subsection support these claims.

As usual, our numerical calculations are "twin experiments" in which for a selected D we choose $\mathbf{x}(t_0) \equiv \mathbf{x}(0)$. We use a time step $\Delta t = 0.025$ to generate solutions $\mathbf{x}(t)$ over an observation window $[t_0, t_{\text{final}}]$ in which $t_0 \leq t \leq t_0 + M\Delta t = t_{\text{final}}$. To each $x_a(t)$, we add Gaussian noise with mean zero and variance $\sigma^2 = 0.25$; these now comprise our library of observed data. We then select $L \leq D$ of these noisy data, and the action reads

$$\begin{aligned} A(\mathbf{X}) = &\sum_{k=0}^{F}\sum_{\ell=1}^{L}\frac{R_m}{2}\left[x_\ell(\tau_k) - y_\ell(\tau_k)\right]^2 \\ &+ \sum_{m=0}^{M-1}\sum_{a=1}^{D}\frac{R_f}{2}\left[x_a(m+1) - f_a(\mathbf{x}(m),\nu)\right]^2. \end{aligned} \tag{3.7}$$

The calculations are performed with the following choices: $D = 20$, $\alpha = 1.4$, $R_{f_0} = 4.0$, $R_m = 4.0$, $N_I = 50$, $\Delta t = 0.025$, and $L = 5, 12$.

In Fig. 3.10 we display the action levels as a function of R_f at $L = 5$. We can see that the Random-proposal Monte Carlo identifies many action levels, corresponding to many peaks in the conditional probability distribution $\pi(\mathbf{X}\,|\,\mathbf{Y}) \propto \exp[-A(\mathbf{X})]$ with the action given in Eq. (3.7). From $R_f = 2 \times 10^4$, we see one level moving away from the collection of larger action levels as β increases. However, no action level has become essentially independent of R_f suggesting that the accuracy with which the model is enforced

remains too small. We expect that as the number of measurements at each τ_k is increased more information about the phase space instabilities will be passed from the data to the model and that the structure of the action level plot will change.

In Fig. 3.11, we now display the action levels and its components, the measurement errors and the model errors, at $L = 12$. Here the behavior of the action levels is quite different. The model error decreases over a large range of R_f until the numerical stability of the evaluation of this term is reduced as small errors in $\mathbf{x}(m+1) - \mathbf{f}(\mathbf{x}(m), \nu)$ are magnified by large values of R_f. As this result appears, the action for each of the N_I paths at each β levels off, becoming essentially independent of R_f. However, the plateaued action level is significantly higher than the expected value calculated from the χ^2 distribution. This indicates that the quality of the transfer of information is sub-optimal compared to that of the PAHMC method.

Until this point, we have examined outcomes of the Precision Annealing Random-proposal Monte Carlo procedure. All of the state variables (*observed* and *unobserved*) as well as the forcing parameter were reported over the observation window $0 \leq t \leq 5$. In a "twin experiment" as in here, we have generated the data by solving a known dynamical equation and adding noise to the output of the $D = 20$ times series with a known value of ν. The point of a twin experiment is to test the method of transfer of information. As we have $D - L$ unobserved state variables at each L, and an unobserved parameter ν, the only tool to determine how well the estimation procedure has done in its task is to predict for $t > 5$ into a prediction window where no information from observations is passed back from the model. We now examine how well the estimation has been performed by predicting both an observed and an unobserved time series among the D available. The points in this paragraph have already been demonstrated extensively in the PAHMC results in Sec. 3.1.2 through Sec. 3.1.5.

We already see from Fig. 3.11(c) that the input value of $\nu = 8.17$ has not been

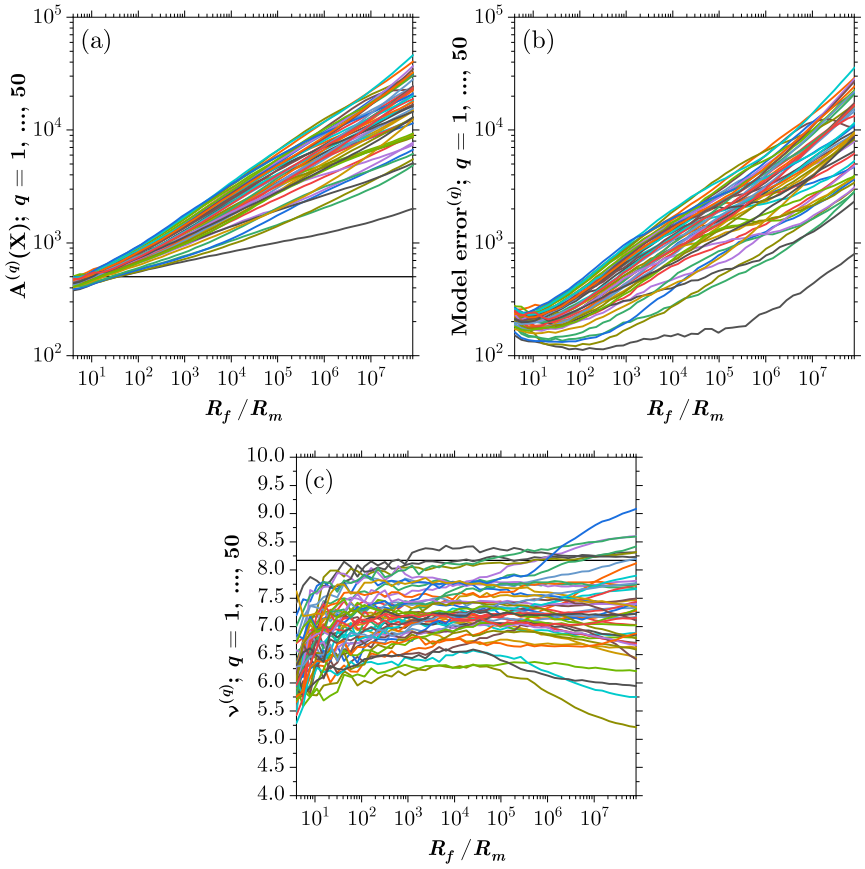

Figure 3.10: Action, measurement error, and model error for Random-proposal Monte Carlo on the Lorenz96 model with $D = 20$ and $L = 5$. Each step in the Precision Annealing procedure is associated with one value of R_f. We perform $N_I = 50$ independent calculations starting at N_I different $(\mathbf{X}(0), \mathbf{P}(0))$. (a) The values of the actions as in Eq. (3.7). The actions are evaluated at $\beta = \log_\alpha(R_f/R_m)$ where $\alpha = 1.4$ and $R_m = 4.0$. Displayed here are the $N_I = 50$ action values at each R_f. These actions are evaluated along the expected path resulting from the sampled paths generated during the Random-proposal procedures from each of the N_I initial paths. (b) The values of the model errors. (c) The values of the forcing parameter.

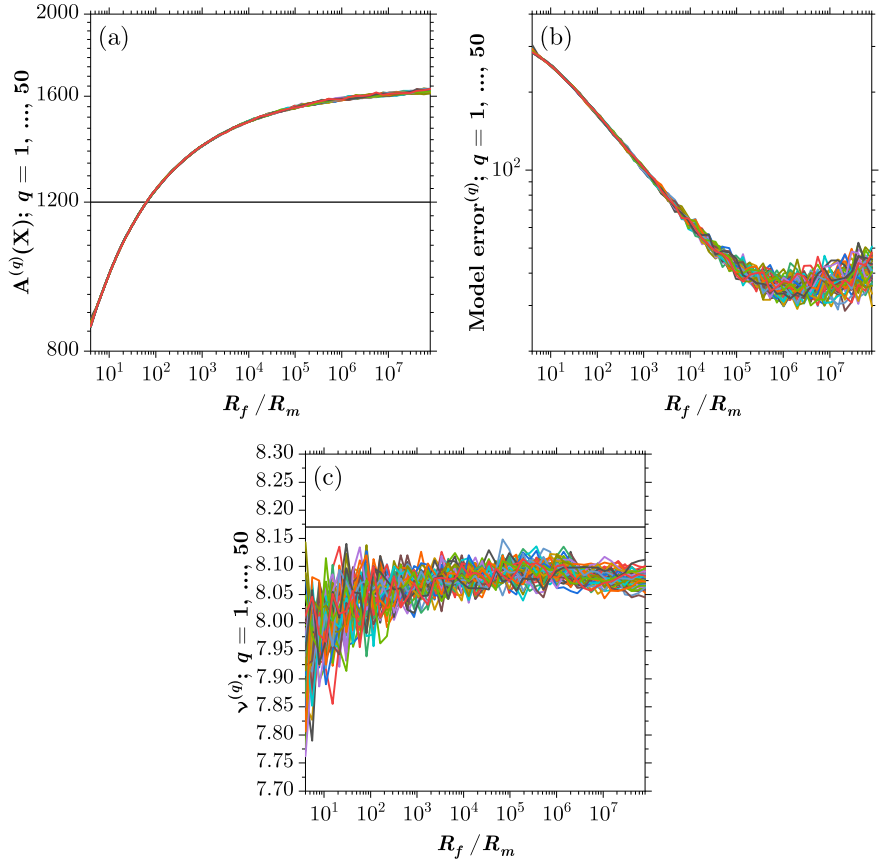

Figure 3.11: Action, measurement error, and model error for Random-proposal Monte Carlo on the Lorenz96 model with $D = 20$ and $L = 12$. Each step in the Precision Annealing procedure is associated with one value of R_f. We perform $N_I = 50$ independent calculations starting at N_I different $(\mathbf{X}(0), \mathbf{P}(0))$. (a) The values of the actions as in Eq. (3.7). The actions are evaluated at $\beta = \log_\alpha(R_f/R_m)$ where $\alpha = 1.4$ and $R_m = 4.0$. Displayed here are the $N_I = 50$ action values at each R_f. These actions are evaluated along the expected path resulting from the sampled paths generated during the Random-proposal procedures from each of the N_I initial paths. (b) The values of the model errors. (c) The values of the forcing parameter.

estimated accurately. The apparent bias in this parameter estimation has also been seen in an earlier Monte Carlo twin experiment [43, 79]. However, note that the PAHMC method can accurately estimate the parameter ν without an apparent bias as shown in Fig. 3.11.

Fig. 3.12(a) shows the results for an observed model variable $x_2(t)$. The observed dimensions are $\ell \in \{1, 2, 4, 6, 7, 9, 11, 12, 14, 16, 17, 19\}$. The estimation of $x_2(t)$ during the observation window using the Random-proposal Monte Carlo is shown in red, and the prediction using all the estimated states of the model at $\mathbf{x}(t = 5)$ and the estimated model parameter is shown in blue. Our knowledge of this dynamical system indicates that the largest global Lyapunov exponent is approximately 1.2 in the time units indicated by Δt. The deviation of the predicted trajectory $x_2(t)$ from the data near $t \approx 6.0$ is acceptable given this Lyapunov exponent. However, this prediction result is significantly worse than that of the PAHMC method (also at $L = 12$).

Fig. 3.12(b) shows an unobserved model variable $x_{20}(t)$ from the same set of calculations. The estimation of $x_{20}(t)$ during the observation window using the Random-proposal Monte Carlo is shown in red, and the prediction using all the estimated states of the model at $\mathbf{x}(t = 5)$ and the estimated model parameter is shown in blue. Our knowledge of this dynamical system indicates that the largest global Lyapunov exponent is approximately 1.2 in the time units indicated by Δt. The deviation of the predicted trajectory $x_{20}(t)$ from the data near $t \approx 6.0$ is acceptable given this Lyapunov exponent. However, as in the observed-variable case mentioned above, this prediction result is significantly worse than that of the PAHMC method (also at $L = 12$).

All the results in this subsection shows that the Random-proposal Monte Carlo method is a good candidate for the information transfer problem. But PAHMC is a noticeably superior method, especially when the problem is complicated enough.

In the next section, we will again focus on the performance of the PAHMC method, this time exclusively.

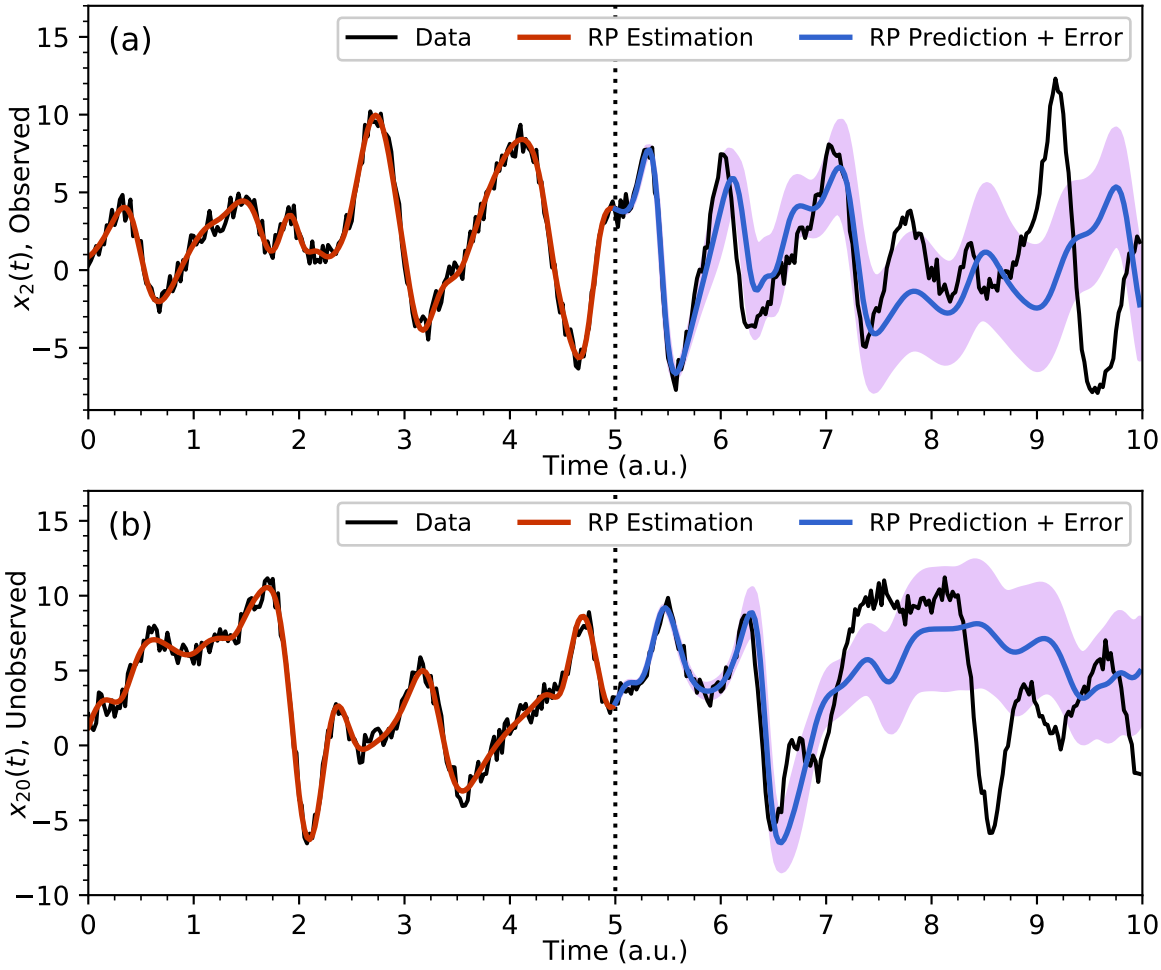

Figure 3.12: Prediction results using Random-proposal Monte Carlo for the Lorenz96 model with $D = 20$ and $L = 12$. The predictions use the values of $\mathbf{x}(t = 5)$ for the full estimated state at the end of the observation window as well as the parameter ν estimated in the Precision Annealing procedure. (a) Results for an observed dynamical variable $x_2(t)$. In black is the full length of the noisy data. In red is the estimated $x_2(t)$ over the observation window $0 \leq t \leq 5$, and in blue is the predicted $x_2(t)$ over the prediction window $5 < t \leq 10$. The prediction error band are in magenta. (b) Results for an unobserved dynamical variable $x_{20}(t)$. In black is the full length of the noisy data. In red is the estimated $x_{20}(t)$ over the observation window (in which the variable is still unobserved), and in blue is the predicted $x_{20}(t)$ over the prediction window. The prediction error band are in magenta.

3.2 The Hodgkin–Huxley Neuron Model

Biophysics is another field where the information transfer problem is ubiquitous and also important. In a biophysical problem, an experimenter is often interested in understanding the properties of a single or a network of cellular or intracellular particles. However, a satisfactory set of measurements to unveil these properties is almost always out of reach due to the limitations of measuring devices, or simply, the inaccessibility of the objects themselves that are being measured [80]. In addition, biophysical experiments usually yield very noisy results.

Nonetheless, a lot of biophysical systems can be appropriately formulated as (nonlinear) dynamical equations, which means we can study them using tools familiar to a statistical physicist. With this in mind, it is therefore possible to extract more information from the measurements than it is possible from solely analyzing the experimental data. This section gives an example.

In this section, we study the application of PAHMC on a well-known model with a fair amount of complexity: the Hodgkin-Huxley model. In 1952, Alan Hodgkin and Andrew Huxley proposed a model in a seminal work that describes the ionic dynamics of transmembrane action potentials in the squid giant axon [21].

Networks of neurons exhibit complex biological and dynamical behaviors, most noticeably, rhythmic bursting and patterned sequence generation [81, 82, 83, 84]. From the perspective of the discussion here, neurons are nonlinear oscillators with competing feedback mechanisms among the flow of ions across the cell membrane. A neuron's cross-membrane voltage depends on currents that flow across various ion channels. In particular, Na^+ has a higher concentration outside the cell and therefore has an inflow tendency. In contrast, K^+ has a higher concentration inside the cell and therefore has an outflow tendency. When there is no outside stimulating signal (which is required for a neuron to "fire"), a neuron maintains a resting potential that is -65mV relative to its exterior, which is a result of the

balance between ion diffusion due to differing concentration and the electrostatic forces on the charged ions [1].

When the neuron is being driven by an external current (i.e., a signal), as the potential passes a threshold, the neuron produces a spike in voltage called the action potential. The existence of the external current implies that we are dealing with a driven dynamical system, which means the dynamical equations will be time-dependent. We will turn to this shortly.

We now turn to the equations under study. The Hodgkin-Huxley (HH) model is the first to quantitatively describe the above process. Following its introduction, the HH model has been studied extensively by physicists and applied mathematicians as a test bed for their new approaches [43, 81, 85, 86, 82, 83, 84].

The HH model consists of dynamical equations for four state variables: the action potential (voltage) $V(t)$ across the cell membrane and the three gating variables $m(t)$, $h(t)$, and $n(t)$. The model describes the dynamics of the voltage and voltage-dependent conductivities. The time evolution of the states are governed by the following first-order differential equations:

$$\begin{aligned} C\,\frac{dV(t)}{dt} &= g_{\mathrm{Na}}\, m(t)^3\, h(t)\,(E_{\mathrm{Na}} - V(t)) \\ &\quad + g_{\mathrm{K}}\, n(t)^4\,(E_{\mathrm{K}} - V(t)) \\ &\quad + g_{\mathrm{L}}\,(E_{\mathrm{L}} - V(t)) \\ &\quad + I_{\mathrm{inj}}(t), \\ \frac{dm(t)}{dt} &= \frac{\eta_m(V(t)) - m(t)}{\tau_m(V(t))}\,, \\ \frac{dh(t)}{dt} &= \frac{\eta_h(V(t)) - h(t)}{\tau_h(V(t))}\,, \\ \frac{dn(t)}{dt} &= \frac{\eta_n(V(t)) - n(t)}{\tau_n(V(t))}\,. \end{aligned} \tag{3.8}$$

In Eq. (3.8), $I_{\text{inj}}(t)$ is the external current applied to the neuron, and the η's and τ's are expressed as

$$\begin{aligned}
\eta_m(V(t)) &= \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{V(t)-V_m}{\Delta V_m}\right), \\
\tau_m(V(t)) &= \tau_{m0} + \tau_{m1}\left[1-\tanh^2\left(\frac{V(t)-V_m}{\Delta V_m}\right)\right], \\
\eta_h(V(t)) &= \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{V(t)-V_h}{\Delta V_h}\right), \\
\tau_h(V(t)) &= \tau_{h0} + \tau_{h1}\left[1-\tanh^2\left(\frac{V(t)-V_h}{\Delta V_h}\right)\right], \\
\eta_n(V(t)) &= \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{V(t)-V_n}{\Delta V_n}\right), \\
\tau_n(V(t)) &= \tau_{n0} + \tau_{n1}\left[1-\tanh^2\left(\frac{V(t)-V_n}{\Delta V_n}\right)\right].
\end{aligned} \tag{3.9}$$

Eqs. (3.8) and (3.9) collectively specify a dynamical system with four state variables $V(t)$, $m(t)$, $h(t)$, and $n(t)$, and a set of nineteen parameters $\boldsymbol{\theta} = \{C, g_{\text{Na}}, E_{\text{Na}}, g_{\text{K}}, E_{\text{K}}, g_{\text{L}}, E_{\text{L}}, V_m, \Delta V_m, \tau_{m0}, \tau_{m1}, V_h, \Delta V_h, \tau_{h0}, \tau_{h1}, V_n, \Delta V_n, \tau_{n0}, \tau_{n1}\}$.

The task of information transfer is to estimate the voltage $V(t)$, all three gating variables $m(t)$, $h(t)$, and $n(t)$, as well as the parameter set $\boldsymbol{\theta}$. Among all these, the most realistic assumption is that only a noisy version of the cross-membrane voltage $V(t)$ is known to us. Of course, the applied current $I_{\text{inj}}(t)$ is known as well. All other state variables and the parameters are all unobserved and need to be estimated.

Once we have transferred information from the $V(t)$ measurements to the HH model specified in Eqs. (3.8) and (3.9), we can use the completed model to predict the response of the neuron for $t > t_{\text{final}}$. If the predictions are accurate for a broad range of biologically plausible $I_{\text{inj}}(t)$, then we know that the PAHMC method is a reliable tool for interpreting and predicting the behavior of the neuron. To assess the quality of the transfer of information, we can always use a validation common in statistical learning.

3.2.1 Data Preparation

Here, we present the details on how the twin experiment data (and the validation set) are generated. These include the true parameters $\boldsymbol{\theta}_{\text{true}}$, the external current $I_{\text{inj}}(t)$, the generated data, and more.

As discussed above, the HH model has $D = 4$. Since only $V(t)$ is observable, we have $L = 1$. We choose $\Delta t = 0.02$ms as the discretization interval for integrating the HH model. The discretization rule is Eq. (3.6), same as what is used in the previous Lorenz96 example. The observation window runs from $t_0 = 0$ to $t_{\text{final}} = 100$ms. Therefore, $M = 5000$. The prediction window is $100\text{ms} < t < 1000\text{ms}$. For the observed dimension $V(t)$, for each of the M data points, we add a Gaussian noise with mean zero and $\sigma^2 = 1.0$.

Table 3.1 records the true values of the parameters used to generate data. The membrane capacitance is held fixed as $C = 1.0\mu\text{F/cm}^2$ and is not subject to estimation.

Table 3.1: The true parameters of the Hodgkin–Huxley model.

Parameter	True Value	Unit
g_{Na}	120.0	mS/cm^2
E_{Na}	50.0	mV
g_{K}	20.0	mS/cm^2
E_{K}	-77.0	mV
g_{L}	0.3	mS/cm^2
E_{L}	-54.4	mV
V_m	-40.0	mV
ΔV_m	15.0	mV
τ_{m0}	0.1	ms
τ_{m1}	0.4	ms
V_h	-60.0	mV
ΔV_h	-15.0	mV
τ_{h0}	1.0	ms
τ_{h1}	7.0	ms
V_n	-55.0	mV
ΔV_n	30.0	mV
τ_{n0}	1.0	ms
τ_{n1}	5.0	ms

Fig. 3.13 shows the simulated applied current, $I_{\text{inj}}(t)$, on the HH model. The univariate time series shown is the first dimension $x(t)$ of the solution to the Lorenz63 dynamical system [18] with $\rho = 28$, $\sigma = 10$, and $\beta = 8/3$. When solving the system, the time is "diluted" by a factor of 20, and the data are re-mapped onto a discretization interval of $\Delta t = 0.02$ms.

We now use $\mathbf{f}(\mathbf{x}(k), \boldsymbol{\theta})$ to denote the model in discrete time, in which $\mathbf{x}(t) = (V(t), m(t), h(t), n(t))$, and $\boldsymbol{\theta}$ is the collection of model parameters. Given that we can only measure $V(t)$ and that we have observation at each time step, i.e., $n_k = k$, the action can be written as

$$A(\mathbf{X}) = \frac{R_m}{2M} \sum_{k=1}^{M} \left[V(k) - y(k)\right]^2 + \sum_{a=1}^{4} \sum_{k=1}^{M-1} \frac{R_f^{(x_a)}}{2M} \left[x_a(k+1) - f_a(\mathbf{x}(k), \boldsymbol{\theta})\right]^2 , \qquad (3.10)$$

where $a \in \{V(t), m(t), h(t), n(t)\}$ is the generic notation for the state variables.

Fig. 3.14 shows the data generated using all the specifications discussed here. The only observed dimension is $V(t)$, and it is noisy. The three gating variables, $m(t)$, $h(t)$, and $n(t)$, are unobserved and need to be estimated by PAHMC.

Besides the three gating variables, we also need to estimate the observed $V(t)$ within the observation window $0 < t < 100$ms, plus the eighteen parameters, in order to complete the HH model specified by Eqs. (3.8) and (3.9). Once the process is finished, we can validate the PAHMC model by (1) comparing the estimations with the ground truth in Fig. 3.14 within the observation window (2) predicting forward for $100\text{ms} < t < 1000$ms and comparing the predictions with the ground truth. If the results are satisfactory, we say that the transfer of information is successful.

The total dimensionality of the problem is therefore $D \times M + 18 = 20018$. This number poses a serious challenge to PAHMC. In addition, the nonlinearities in Eqs. (3.8) and (3.9) makes the action manifold specified by Eq. (3.10) highly nonconvex.

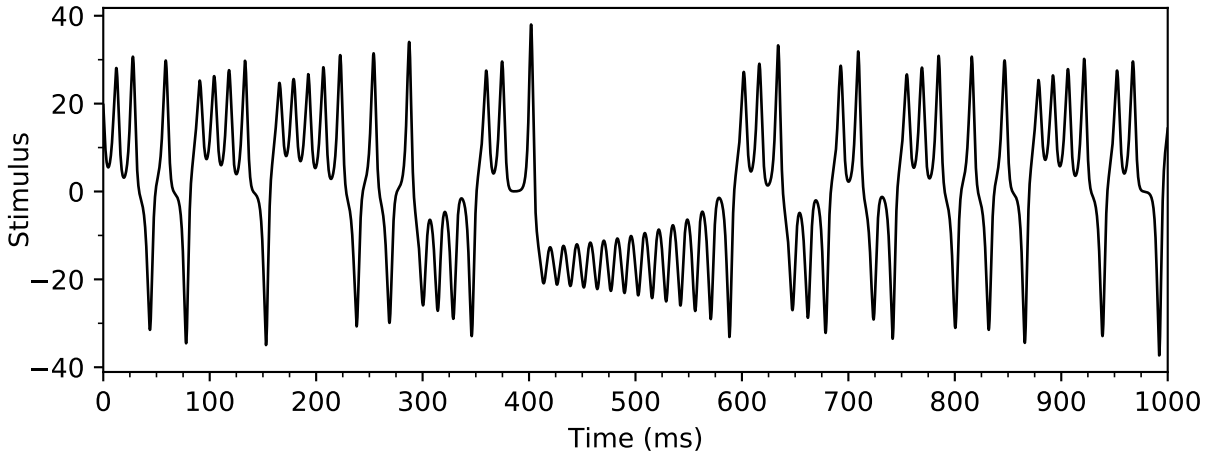

Figure 3.13: The external stimulus current, $I_{\text{inj}}(t)$, applied on the Hodgkin–Huxley model. The time series is the first dimension $x(t)$ of the solution to the Lorenz63 dynamical system. When solving the Lorenz63 system, the discretization interval (in a.u.) used is 0.001. The data are then re-mapped onto a $\Delta t = 0.02$ms interval.

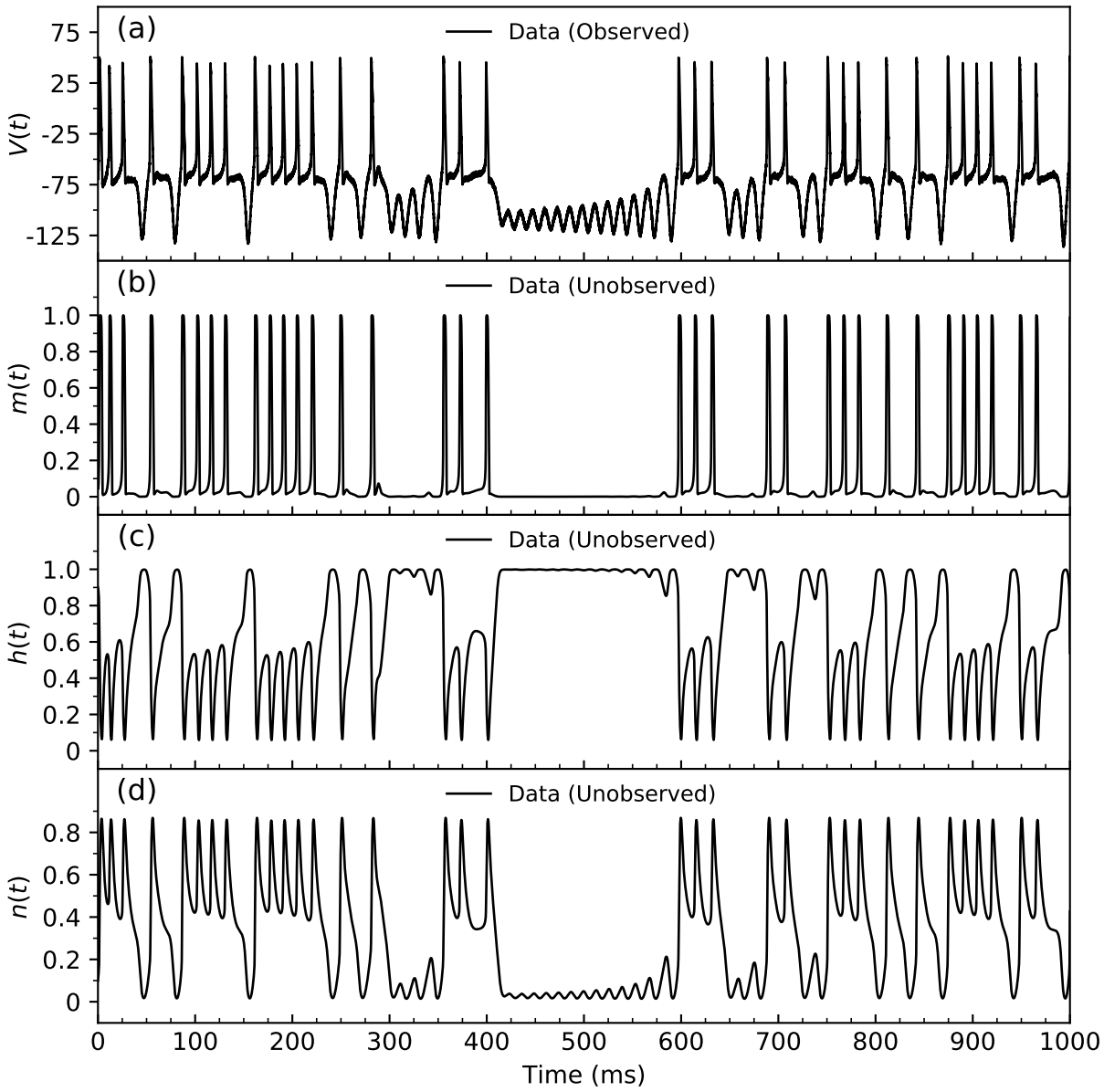

Figure 3.14: The generated data for the Hodgkin–Huxley neuron model. $V(t)$ is observed and has noise. The three gating variables, $m(t)$, $h(t)$, and $n(t)$ are unobserved and need to be estimated.

3.2.2 Results from PAHMC

We present here the results of the transfer of information from the $V(t)$ data into the HH model described in Eqs. (3.8) and (3.9). The external current $I_{\text{inj}}(t)$ is provided throughout the observation and prediction window.

Essentially, PAHMC performs sampling from the high-dimensional distribution $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$, identifies the dominant contribution to the expected-value integral in Eq. (2.11), and gives an accurate estimation of $\langle \mathbf{X} \rangle$ and higher moments. The final product of this process contains all the information needed to estimate $V(t)$, the unobserved gating variables $m(t)$, $h(t)$, and $n(t)$, and the eighteen parameters. One can then make predictions based on these estimations.

Before showing the numerical results, we first report how Precision Annealing (PA) is performed in the calculations for the Hodgkin-Huxley model. PA was discussed in great detail in Chapter 2 as a step-wise procedure to guide HMC to the region in the $\mathbf{X}$ space where the majority of the probability mass is located.

In Eq. (3.10), the for of the action differs slightly from the original definition in Eq. (2.13) such that we allow each of the D state variables to have a unique R_{f_0} value. This treatment brings no conceptual difference but is beneficial for numerical calculations when different state variables have different dynamical ranges, as is the case for the Hodgkin-Huxley model. In other words, we have

$$R_f^{(x_a)} = R_{f_0}^{(x_a)} \times \alpha^{\beta}, \tag{3.11}$$

where $a \in \{V(t), m(t), h(t), n(t)\}$ is the generic notation for the state variables. For the calculations in this section, we choose $\alpha = 2.0$ and $\beta = 0, \ldots, 19$.

As a rule of thumb, the value of R_{f_0} for a state variable should be inversely proportional to the square of its corresponding dynamical range. This is because we want

the shape of the $A(\mathbf{X})$ manifold in the $\mathbf{X}$ space to be as "regular" as possible. Specifically, the dynamical range for $V(t)$ is roughly $(-125, 50)$ and dynamical range for the three gating variables $m(t)$, $h(t)$, and $n(t)$ is $(0, 1)$. Our choices for the R_{f_0}'s are therefore $R_{f_0}^{(V)} = 0.1$, $R_{f_0}^{(m)} = 1200$, $R_{f_0}^{(h)} = 1600$, and $R_{f_0}^{(n)} = 2100$. The smallest value, 0.1, ensures that we impose a weak enough model constraint on the action at the beginning.

Fig. 3.15 plots the action and the model error resulted from PAHMC calculations. We can see from the two panels that the transfer of information is quite successful because the action quickly plateaus to the asymptotic value calculated from the expectation of the χ^2 distribution associated with the noise in $V(t)$.

Table 3.2 records the estimated values of the eighteen model parameters by PAHMC. These accurate estimations of parameters, along with ideal action level and model error, constitute a good basis for making predictions.

Table 3.2: The estimated parameters of the Hodgkin–Huxley model.

Parameter	True Value	Estimated	Unit
g_{Na}	120.0	116.19	$\mathrm{mS/cm^2}$
E_{Na}	50.0	49.72	mV
g_{K}	20.0	21.63	$\mathrm{mS/cm^2}$
E_{K}	-77.0	-77.05	mV
g_{L}	0.3	0.30	$\mathrm{mS/cm^2}$
E_{L}	-54.4	-54.27	mV
V_m	-40.0	-40.18	mV
ΔV_m	15.0	14.80	mV
τ_{m0}	0.1	0.10	ms
τ_{m1}	0.4	0.40	ms
V_h	-60.0	-59.34	mV
ΔV_h	-15.0	-14.09	mV
τ_{h0}	1.0	0.99	ms
τ_{h1}	7.0	8.78	ms
V_n	-55.0	-53.92	mV
ΔV_n	30.0	31.76	mV
τ_{n0}	1.0	1.03	ms
τ_{n1}	5.0	4.94	ms

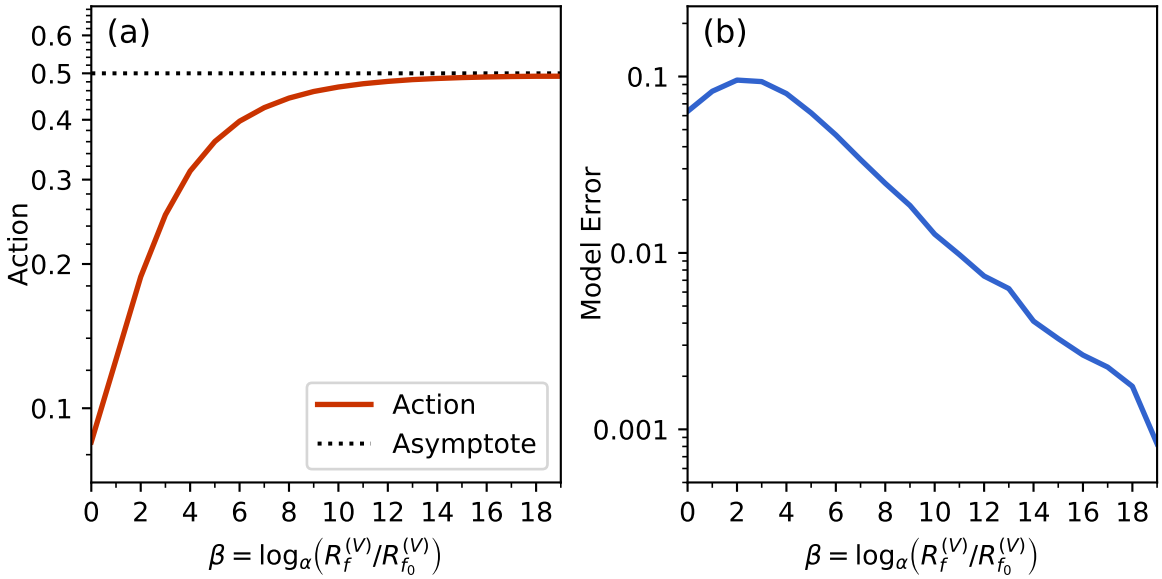

Figure 3.15: Action level and model error for PAHMC on the Hodgkin-Huxley model with $D = 4$, $L = 1$, and $M = 5000$. Each step in the Precision Annealing Procedure is associated with one value of β. Each of the four state variables has a different R_{f_0} value. With only one initial condition, i.e., $N_I = 1$, we can still conclude that the information transfer is successful. (a) Action values at each β. The action quickly plateaus as β grows. (b) The model error in Eq. (3.10) as a function of β.

Following the action and parameter-estimation results, we are now ready to discuss the estimation and prediction of states. Fig. 3.16 shows the estimation and prediction results on the observed state variable, $V(t)$, as well as on the three unobserved gating variables, $m(t)$, $h(t)$, and $n(t)$. Within the observation window $0 < t < 100$ms, all four estimations are accurate despite $V(t)$ being noisy and $m(t)$, $h(t)$, and $n(t)$ being "hidden" to PAHMC. This, combined with the fact that the parameter estimations are accurate as shown in Table 3.2, indicate that the principal mode of the probability distribution $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$ has been found and that the information has been successfully transferred into the Hodgkin-Huxley model.

To further solidify the claim of success, we can look at the predictions of the four state variables within 100ms $< t <$ 1000ms in Fig. 3.16. We see that all of the predictions for $V(t)$ and for $m(t)$, $h(t)$, and $n(t)$ align perfectly with the data in the validation set. In addition, the error bands are plotted, but they are barely visible since PAHMC is confident about its predictions. This phenomenon of narrow error bands is a result of both the variance-reduction capability of HMC and the fact that most of the probability mass of $\pi(\mathbf{X} \,|\, \mathbf{Y})$ is located around the proximity of the global minimum of $A(\mathbf{X})$.

At this point, we have presented comprehensive evidence of the efficacy of PAHMC as a method to transfer information from partial, noisy data to a dynamical system. In Sec. 3.1 we discussed PAHMC results on the $D = 20$ Lorenz96 chaotic system with $L = 7, 8, 10, 12$. We saw that about 40% of the state variables need to be observed in order to achieve a satisfactory transfer of information. In this section, we reported PAHMC results on the non-chaotic but much more complicated Hodgkin-Huxley model. Among the four state variables, only $V(t)$ needs to be observed. We validated the PAHMC results by comparing the estimations and predictions with the ground truth.

We conclude that the PAHMC method proposed in this dissertation is effective, and it can be applied to a broad range of problems.

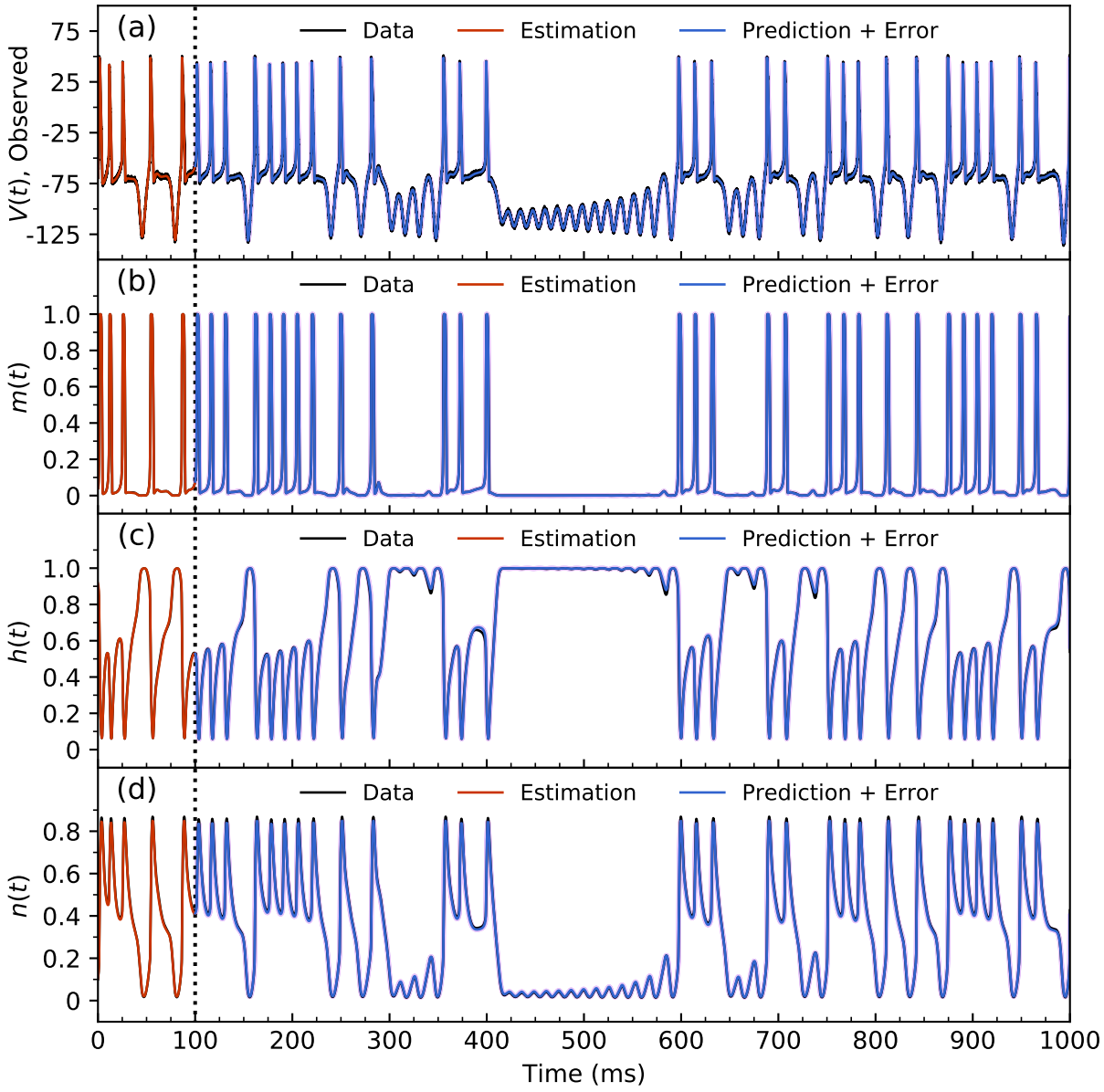

Figure 3.16: Predictions by PAHMC on the Hodgkin-Huxley model with $D = 4$, $L = 1$, and $M = 5000$. (a) Display of results for the observed state variable $V(t)$. The noisy data span the full $0 < t < 1000$ms interval. The estimated values in the observation window $0 < t < 100$ms are shown in red. The predicted values, with errors (in magenta), are shown in blue within the interval of 100ms $< t <$ 1000ms. (b) Display of results for the unobserved gating variable, $m(t)$. (c) Display of results for the unobserved gating variable, $h(t)$. (d) Display of results for the unobserved gating variable, $n(t)$.

3.3 Python Implementations

The Python implementations of the PAHMC method are available on the author's GitHub web page.

The CPU version is available at

`https://github.com/zfang92/pahmc-ode-cpu`.

The GPU version is available at

`https://github.com/zfang92/pahmc-ode-gpu`.

Chapter 4

Discussion

We formulated the theory of the Precision Annealing Hamiltonian Monte Carlo (PAHMC) method in Chapter 2 and presented numerical examples regarding its applications in Chapter 3. The current chapter serves to address some important questions and to summarize the key contributions.

In Sec. 4.1 we discuss the necessity of dealing with the path $\mathbf{X}$, which is by definition a high-dimensional quantity. In other words, *why* we must use the state-space representation to complete the task of information transfer.

Then, in Sec. 4.2, we cover the computational complexity of the PAHMC method as a guide to its scaling properties in complicated problems.

We conclude the chapter with a brief summary.

4.1 Why State Space?

Recall that the path $\mathbf{X}$ is defined in Eq. (2.4) as $\mathbf{X} = \big\{\mathbf{x}(0), \ldots, \mathbf{x}(M); \theta_1, \ldots, \theta_{N_p}\big\}$. The number of degrees of freedom of $\mathbf{X}$ is therefore $D(M+1) + N_p$. For the Lorenz96 system discussed in Sec. 3.1, this number exceeds 4000. For the Hodgkin-Huxley model subsequently discussed in Sec. 3.2, this number exceeds 20000.

At this point, two important questions on the necessity to have a "heavy" object $\mathbf{X}$ may be raised.

(1) Why do we need to estimate both the *observed* and *unobserved* degrees of freedom in $\mathbf{X}$? Taken to an extreme, if all of $\mathbf{X}$ can be observed, can we only estimate the parameter set $\boldsymbol{\theta}$ and then use this and the available $\mathbf{Y}$ to make predictions?

(2) Can we manage to work with the space spanned by $\{\mathbf{x}(0), \boldsymbol{\theta}\}$, i.e., the space that only contains the initial conditions and the parameter set $\boldsymbol{\theta}$? If so, the total degrees of freedom that we need to deal with would drop dramatically from $D(M+1)+N_p$ to $D+N_p$. In other words, this idea is equivalent to *not* having model errors in the action.

We answer these questions in the following subsections, respectively. Before doing so, let us first review the notation and define a couple of new ones.

The state-space representation is widely adopted by many fields such as numerical weather prediction and control theory [2, 87, 88]. In Sec. 2.2, Eq. (2.1) establishes the dynamical model in continuous time. Since we always work in discrete time for any model that is not exactly solvable, we arrive at Eq. (2.2), which states $x_a(m+1) = f_a(\mathbf{x}(m), \boldsymbol{\theta})$, where $\mathbf{x}(m) = (x_1(m), \ldots, x_D(m))$ is the state vector at discrete time mark m. The index a ranges from 1 to D and time m ranges from 0 to M.

Assuming we have in total N_p parameters in $\boldsymbol{\theta}$, the collection of all the state variables, or simply, the path $\mathbf{X}$, has

$$\mathcal{T} \equiv D(M+1) + N_p \tag{4.1}$$

total degrees of freedom.

Now, given that we can only observe L out of D dimensions of the dynamics, and, without loss of generality, assuming that we can observe the system at each time $m = 0, \ldots, M$, our (noisy) data can then be denoted as $\mathbf{Y} = \{\mathbf{y}(0), \ldots, \mathbf{y}(M)\}$, where each $\mathbf{y}$ is an L-dimensional vector. There are thus $L(M+1)$ degrees of freedom in $\mathbf{Y}$.

As a result, there are clearly

$$\mathcal{K} \equiv (D - L)(M + 1) + N_p \tag{4.2}$$

degrees of freedom "missing." And these $\mathcal{K}$ degrees of freedom make up the *unobserved* portion of $\mathbf{X}$ and need to be estimated.

4.1.1 The Necessity of Estimating Every Degree of Freedom

Here, we address the first question raised above. We show that not only the unobserved $\mathcal{K}$ but also all $\mathcal{T}$ degrees of freedom must be estimated in all cases in order to achieve a meaningful transfer of information from data to model. Again, taking things to an extreme, even if we had only one parameter θ in our model (such as in the Lorenz96 system) and we had a full set of observations such that $L = D$, we may still have thousands of, if not more, variables left to be estimated. This makes the information transfer task far more challenging than fitting only the set of parameters $\boldsymbol{\theta}$.

The first reason is that the data set $\mathbf{Y}$ contains only partial and noisy information on $\mathbf{X}$. Even though the dynamical evolution $\mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$ is deterministic, the data obtained from observing the dynamics are noisy. In many cases, this means $y_a(m) = x_a(m) + \eta_a$ for $a = 1, \ldots, L$, where $\eta_a \sim \mathcal{N}(0, \sigma_a)$. This suggests that every component of $\mathbf{X}$ needs to be estimated.

Even if we could somehow observe all D out of D dimensions at each discrete time mark, i.e., $\mathcal{K} = N_p$ in Eq. (4.2), we still cannot avoid estimating the $D(M + 1)$ degrees of freedom in $\mathbf{X}$. This is because the inherent noise in $\mathbf{Y}$ would produce a large model error in the action given by Eq. (2.13), thus preventing any method from locating the optimal parameter set $\boldsymbol{\theta}^*$.

Suppose in an even more unrealistic scenario we are given the true values of $\boldsymbol{\theta}$ and

a full observation ($L = D$) of $\mathbf{X}$, can we avoid estimating the full degrees of freedom in $\mathbf{X}$ and use only $\{\mathbf{Y}, \boldsymbol{\theta}_{\text{true}}\}$ to make predictions beyond the observation window? The answer is still *No*.

Fig. 4.1(b) gives an example through the Lorenz96 system where the $D = 20$ dimensions are fully observed and the true forcing parameter used to generate the data $\theta_{\text{true}} = 8.17$ is known. Nevertheless, starting from the end of the observation window at $t = 5.0$, the prediction for $x_3(t)$ quickly diverges from the noisy ground truth. Comparing this with the results in Sec. 3.1, which always had $L < D$ and θ_{true} unknown, we can conclude that all the $D(M + 1) + N_p$ degrees of freedom in $\mathbf{X}$ should be estimated before making predictions. This is equivalent to saying that in any circumstance, we cannot bypass the process of transferring information from data to model in order to make meaningful predictions.

4.1.2 The Harsh Geometry of the Initial-Condition Space

Now let us turn to the second question raised at the beginning of this section. We have just discussed the necessity of using the state-space representation for our information transfer task. However, since we work with non-stochastic models for which $\mathbf{x}(m)$ *could* be obtained by forward integration using information on $\mathbf{x}(0)$ and $\boldsymbol{\theta}$, can we avoid working with the $\mathcal{T}$-dimensional space spanned by $\mathbf{X}$? In other words, can we only work with the $(D + N_p)$-dimensional space spanned by $\{\mathbf{x}(0), \boldsymbol{\theta}\}$? We would hope this to be possible, but the answer is, again, *No*.

Specifically, we illustrate that imposing the equality in Eq. (2.2) at the outset and working only with $\mathbf{x}(0)$ and $\boldsymbol{\theta}$ would make it impossible to achieve the goal of transferring information. Consequently, we need to deal with all $\mathcal{T}$ degrees of freedom in $\mathbf{X}$, not only the $D + N_p$ that corresponds to $\mathbf{x}(0)$ and $\boldsymbol{\theta}$.

We know the action consists of a measurement error term and a model error term,

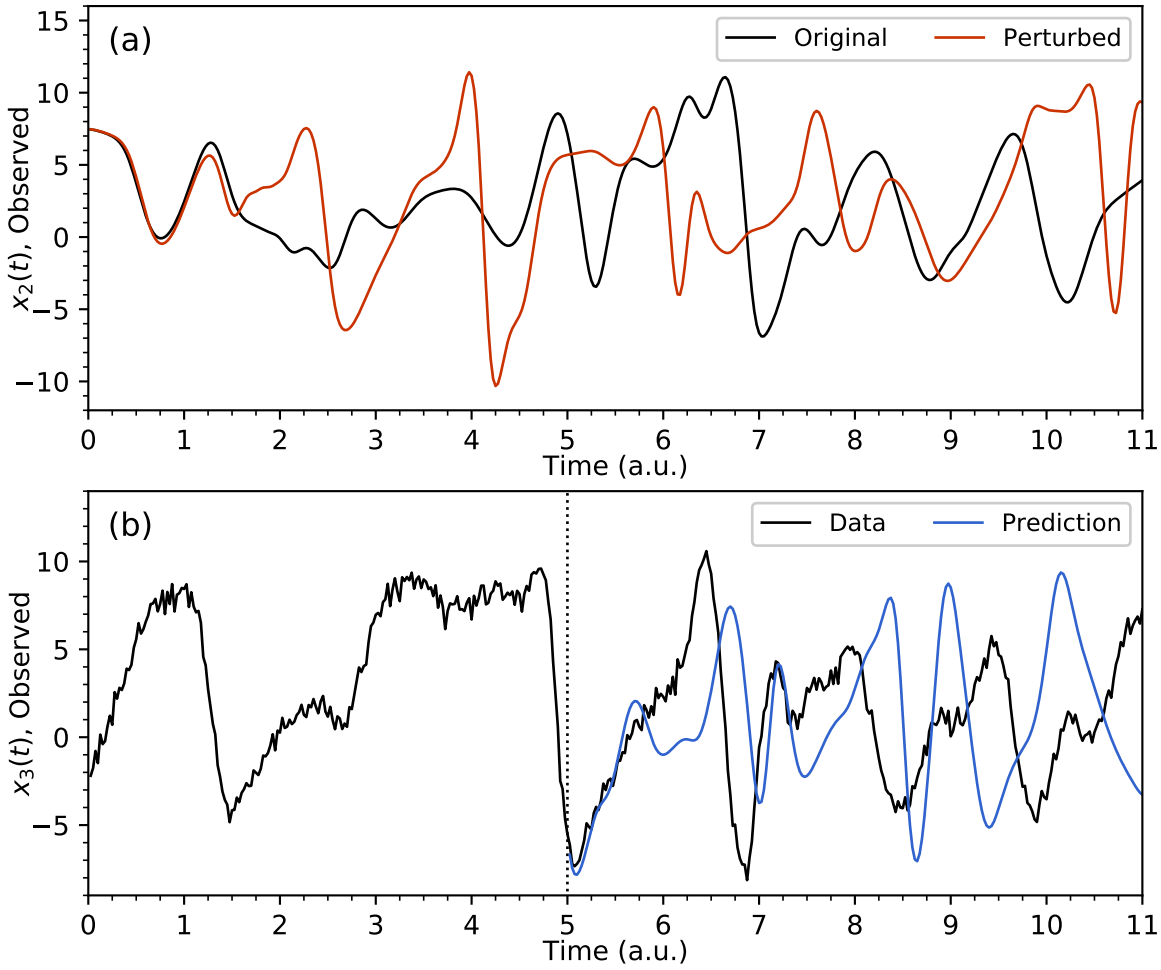

Figure 4.1: Two attempts to forward-integrate a fully observed Lorenz96 system without first completing the transfer of information. These results have no predictive power. (a) Perturbation of the $x_2(t)$ variable of a fully observed, $D = 20$ system with no noise added to the data. Even with an infinitesimal change in the initial condition $\mathbf{x}(0)$, the resulting time series quickly diverges from the ground truth. (b) An attempt to predict a fully observed, $D = 20$ system (with noise in the data) by directly integrating from the end of the observation window. The $x_3(t)$ variable is shown.

as described by Eq. (2.13). However, if we do work in the $\{\mathbf{x}(0), \boldsymbol{\theta}\}$ space, we are forced to impose the equality on the dynamics $x_a(m+1) = f_a(\mathbf{x}(m), \boldsymbol{\theta})$ from the outset. This results in a zero model error in the action, and the whole process of estimating $\mathbf{x}(0)$ and $\boldsymbol{\theta}$ resembles "solving a nonlinear two-point boundary value problem by searching for the right initial condition at the left boundary and comparing the integration results with data to see if they match." This could not work, as illustrated in Fig. 4.1(a) and Fig. 4.2. Geometrically, working in the $\{\mathbf{x}(0), \boldsymbol{\theta}\}$ space means moving about on a $(D + N_p)$-dimensional nonconvex action manifold, hoping to locate the abrupt decline in the action that corresponds to the optimal set of $\mathbf{x}(0)$ and $\boldsymbol{\theta}$. There is no means to achieve this.

Let us return to Fig. 4.1(a). The plot shows the hyper-sensitivity with respect to perturbations on the initial condition $\mathbf{x}(0)$ of the Lorenz96 system. This speaks for the fact that there must be many local minima around the global minimum of the action if we force R_f to be infinity.

Indeed, if we turn to Fig. 4.2, which is a two-dimensional projection of the twenty-dimensional $R_f = \infty$ action manifold, we can see that the global minimum is located at $(0, 0)$ in the unit of perturbation to $x_1(0)$ or $x_{15}(0)$. There are many local minima near $(0, 0)$. And the surface is rather bumpy at coordinates away from the global minimum. We can see it is infeasible to conduct either sampling or optimization on such a rough, two-dimensional surface, and things could only get much worse in the original twenty-dimensional space. Any sampling or optimization method would get "trapped" at where it starts off, and no information could be transferred.

Overall, we have shown in this section that one needs to work with all the $\mathcal{T} = D(M+1)+N_p$ degrees of freedom in $\mathbf{X}$ in order to achieve the goal of transferring information from $\mathbf{Y}$ to the dynamical model. Therefore, the task is naturally high-dimensional and is more challenging than either parameter fitting or dealing with a linear system in its initial-condition space.

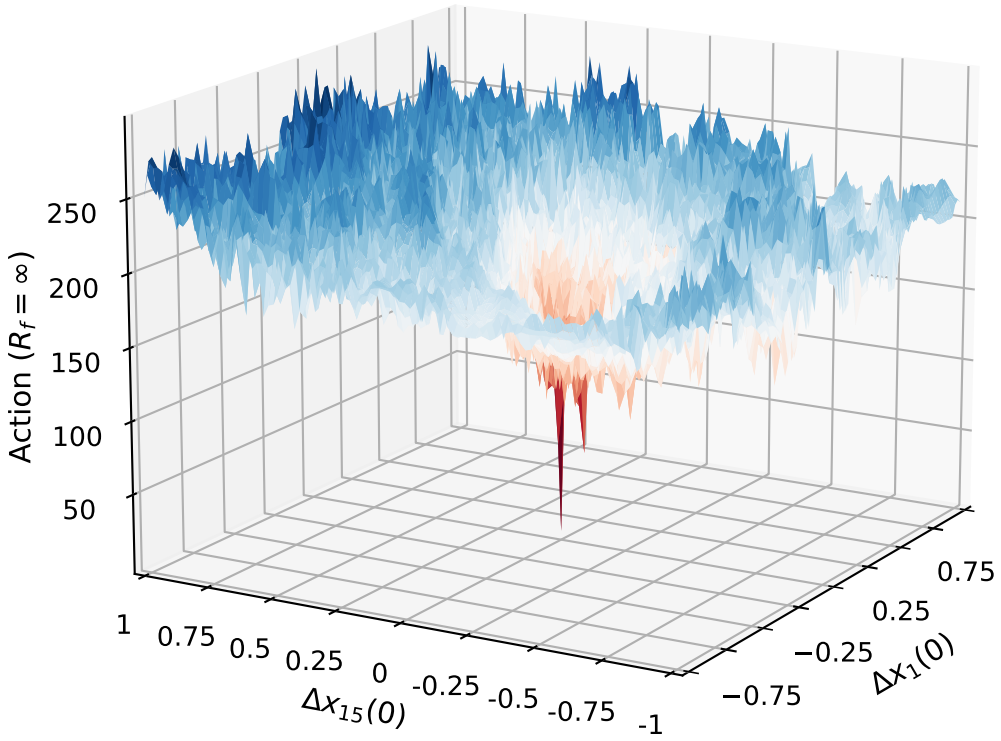

Figure 4.2: A two-dimensional projection of the $R_f = \infty$ action manifold of a $D = 20$ Lorenz96 system. The horizontal axes represent perturbations on $x_1(0)$ and $x_{15}(0)$, respectively.

4.2 Computational Complexity of PAHMC

Having clarified the necessity of working with the high-dimensional object $\mathbf{X}$, we now discuss in this section how computationally efficient the proposed PAHMC method could be.

It is important to consider computational complexity because this is an important factor in real-world practical applications. Equivalently, it is vital that the method achieves good scaling properties when applied to large systems. Here, we give an estimate on the scaling as the system size increases. We first consider the complexity of an elementary step of HMC, which is that of a one-iteration leapfrog simulation. This requires evaluating $\nabla A(\mathbf{X})$ once and therefore has dominant time complexity of $\mathcal{O}(DM)$, where, as usual, D is the dimension of the underlying dynamical system and M is the number of discrete time steps in the observation window. As the dimensionality of the path $\mathbf{X}$ grows, the number of leapfrog steps in one HMC proposal needs to be adjusted accordingly to reach a nearly independent point. In fact, it has been noted [74, 38, 73] that the computation typically grows as the 5/4 power of the dimensions of the model, assuming a near-constant acceptance rate.

As a result, the time complexity of the PAHMC method proposed in this dissertation is $\mathcal{O}((DM)^{5/4})$. Apparently, there exists a multiplicative constant—independent of D and M—that is determined by the choices of the number of samples N_β and the number of Precision Annealing steps $\beta_{\max}$.

The mixing time itself is hard to estimate, yet it has been empirically shown in many cases that HMC achieves faster mixing than other well known Markov chain Monte Carlo methods. In addition, a modification of the term $h(\mathbf{P}, \mathbf{X})$ in the Hamiltonian has been made by Kadakia [64], and it may improve the mixing rate.

It is also useful to provide a benchmark as to the actual computation time for an example system. To this end, we report the time for the Lorenz96 system considered

in this dissertation. In the Lorenz96 example, we are dealing with a 4000-dimensional system for which $D = 20$ and $M = 200$ (ignoring the additional dimension introduced by the model parameter ν). To obtain one HMC proposal, $\mathbf{X}_{\text{prop}}$, we simulate the discrete Hamiltonian dynamics given in Eq. (2.26) for $S = 50$ steps with step size $\varepsilon = 10^{-3}$. In an implementation in Python, the computation time is around 0.02 second. We then make $N_\beta = 10^3$ proposals for every β up to $\beta_{\max} = 30$. As a result, the entire calculation takes about 10 minutes to finish.

To provide a direct comparison with the Precision Annealing Random-proposal Monte Carlo method, we have run in Sec. 3.1.6 a subset of the twin experiments for the Lorenz96 system with $D = 20$ and $M = 200$. For every β up to $\beta_{\max} = 50$, we make 6000 perturbations on $\mathbf{X}$ within each β value. This means we make 2.4×10^7 proposals for each β, given the fact that we only perturb one component of $\mathbf{X}$. Implemented in the C language, the calculation took about four hours to complete. The computation time is much longer than that of the PAHMC method since the Random-proposal method requires far more proposal steps due to inefficient sampling and slow exploration of the $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$ distribution. We have seen in Sec. 3.1.6 that the actual performance of the Random-proposal method lagged that of the PAHMC method in terms of estimation and prediction. Combined with what is discussed here, we say that the PAHMC method is superior in terms of both accuracy and efficiency.

Aside from the above analyses of the $\mathcal{O}((DM)^{5/4})$ rule and the actual computation time, we also point out that the PAHMC method could gain a significant speedup if run in parallel, provided that the action is constructed as in Eq. (2.13). We will again use Lorenz96 to make the point, but it should be noted that the discussion is quite general in the sense that one only needs to substitute the vector field $f_a(\mathbf{x}(m), \boldsymbol{\theta})$ for the system under examination. We focus on the evaluation of $\nabla A(\mathbf{X})$ since this is the only calculation that scales directly with D and M.

First, for the D vector field $\mathcal{F}_a(\mathbf{x}(m), \nu)$ defined in Eq. (3.1) and a discretization rule in time given by Eq. (3.6), the Jacobian for the vector field at $t = t_m$ in the observation window can be written as

$$\begin{aligned} \mathcal{J}_{ij}(m) &= \frac{\partial \mathcal{F}_i(\mathbf{x}(m), \nu)}{\partial x_j(m)} \\ &= \delta_{i-1,j} \left(x_{i+1}(m) - x_{i-2}(m) \right) \\ &\quad + \left(\delta_{i+1,j} - \delta_{i-2,j} \right) x_{i-1}(m) - \delta_{ij} \,, \end{aligned} \tag{4.3}$$

where $i, j = 1, \ldots, D$. The time-consuming part in calculating the D components in $\nabla A(\mathbf{X})$ corresponding to $t = t_m$ is to multiply the Jacobian $\mathcal{J}(m)$ with a D vector. This can be fully parallelized, and a D times speedup can be expected.

Second, to complete the calculation of $\nabla A(\mathbf{X})$, we need to repeat the above process of evaluating the Jacobian for each discrete time mark m from 1 to M. These repetitions are independent of each other and can therefore be fully parallelized. Thus, an M times speedup can be expected.

The massive potential of parallelization discussed above makes PAHMC especially suitable for very large dynamical systems in terms of both the dimensionality of the vector field and the length of the observation window.

4.3 Summary

In this section, we briefly review the key points in Chapters 2 and 3 as well as so far in this chapter.

We have proposed a systematic method to transfer information from observations on a nonlinear dynamical system to a model that describes the time-evolution of the states of that system. The observations are noisy, and the model has errors, so we must use the conditional probability distribution $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$, conditioned on the

observations, as the principal quantity of interest. With this probability, we are then interested in the expected value of any function of $\mathbf{X}$. Generically denoted by $\langle G(\mathbf{X})\rangle$, the expected-value integral is then

$$\langle G(\mathbf{X})\rangle = \frac{\int d\mathbf{X}\, G(\mathbf{X})\, e^{-A(\mathbf{X})}}{\int d\mathbf{X}\, e^{-A(\mathbf{X})}}. \tag{4.4}$$

The *path*, denoted by $\mathbf{X} = \{\mathbf{x}(0), \ldots \mathbf{x}(M), \boldsymbol{\theta}\}$, is the collection of all the state variables in discrete time within the observation window, along with the time-independent model parameters. The approximate numerical evaluation of the expected-value integrals in the form of Eq. (4.4) is one of focuses of this dissertation. Others have also pointed out that the evaluations of expected-value integrals are essential in both data assimilation and deep learning [15, 17].

The hallmark of success in this information transfer process is that one can accurately predict the future evolution of the dynamical system for some time window, called a prediction or generalization interval, beyond the observation window, once given the final estimated state $\mathbf{x}(M)$ and the parameters $\boldsymbol{\theta}$ at the termination of the observation window. Before being able to predict, one must first complete the model $\mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$ by estimating all the degrees of freedom in $\mathbf{X}$. Furthermore, we must work with the full $\mathbf{X}$ instead of the initial-condition space spanned by $\{\mathbf{x}(0), \boldsymbol{\theta}\}$.

Specifically, the observed degrees of freedom in $\mathbf{X}$ must also be estimated. The reason is that when analyzing physical or biophysical systems using the proposed PAHMC method, the estimation of the *observed* degrees of freedom in $\mathbf{X}$ may turn out to be as important as the estimation of the *unobserved* degrees of freedom. In numerical weather prediction [87], for example, good estimates of the full model states of the earth system provides information not necessarily available in the observations themselves. It is the connection of observed and unobserved state variables within the dynamical system that

permits this feature.

On the next level, the work presented in this dissertation develops a systematic way of locating the area around the global minimum of the action $A(\mathbf{X})$ in the path space spanned by $\mathbf{X}$. More importantly, the PAHMC method performs efficiently sampling of the distribution $\pi(\mathbf{X} \,|\, \mathbf{Y}) \propto \exp[-A(\mathbf{X})]$ in order to evaluate the path integral. This accurate sampling of $\pi(\mathbf{X} \,|\, \mathbf{Y})$ enables the estimation of higher moments of $\mathbf{X}$ that can be used in assessing the errors in the prediction.

We used the Hamiltonian Monte Carlo method [37, 38, 39] as the sampling device, and we also make a comparison to the classic, Random-proposal Monte Carlo method based on the Metropolis-Hastings procedure. Both the two-dimensional pedagogical example in Fig. 2.3 and the full results from Sec. 3.1 clearly demonstrated that the HMC method was a far more preferable choice.

HMC was introduced as an innovative version of Monte Carlo sampling for high-dimensional probability distributions. The core idea is to achieve high efficiency by introducing additional degrees of freedom into the target distribution to avoid the random walk-like behavior of the Metropolis-Hastings (MH) procedure. The MH procedure brings lower acceptance rates of proposed moves in the path space as well a markedly slower exploration of the target distribution $\pi(\mathbf{X} \,|\, \mathbf{Y})$.

HMC proceeds by making proposals according to Hamilton's equations for the collection of model state variables $\mathbf{X}$ and their canonical conjugates, $\mathbf{P}$. The HMC samplings occur in the enlarged canonical phase space $(\mathbf{X}, \mathbf{P})$, and the target distribution becomes $\pi(\mathbf{X}, \mathbf{P}) \propto \exp[-H(\mathbf{X}, \mathbf{P})]$ with $H(\mathbf{X}, \mathbf{P}) = A(\mathbf{X}) + h(\mathbf{P}, \mathbf{X})$. A collection of conserved quantities are associated with this method of making proposals. Among them are the phase space volume and $H(\mathbf{X}, \mathbf{P})$ itself. As a result, the overall acceptance rate is close, but not equal, to one.

For a high-dimensional $\mathbf{X}$ and a nonconvex action $A(\mathbf{X})$, HMC itself does not

guarantee that we will capture the set of maxima of the target distribution without additional guidance to find the minima in the action. In fact, from a random initialization point in $\mathbf{X}$, it is unlikely for an HMC sampler to travel for a long distance in the $\mathbf{X}$ space. However, accurate approximation of the path integral in Eq. (4.4) also hinges on locating the global minimum in the action of the form

$$\begin{aligned} A(\mathbf{X}) = & \sum_{k=0}^{F} \sum_{\ell=1}^{L} \frac{R_m}{2(F+1)} \left[x_\ell(\tau_k) - y_\ell(\tau_k) \right]^2 \\ & + \sum_{m=0}^{M-1} \sum_{a=1}^{D} \frac{R_f}{2M} \left[x_a(m+1) - f_a(\mathbf{x}(m), \boldsymbol{\theta}) \right]^2 . \end{aligned} \tag{4.5}$$

To fill this gap, we have presented the Precision Annealing (PA) procedure, which is a development of the early ideas used in the context of variational, optimization-based calculations [44, 45, 47, 48]. The combination of PA and HMC is able to perform the desired expected-value integral with high accuracy.

In the Precision Annealing procedure, R_f in Eq. 4.5 is varied gradually from $R_f \approx 0$ to very large values that strictly enforces the equality in $\mathbf{x}(m+1) = \mathbf{f}(\mathbf{x}(m), \boldsymbol{\theta})$. At $R_f = 0$, the dynamical model is completely unresolved, and the global minimum of the action is therefore easily identified. As one increases R_f, HMC sampling begins at a position in the path space that is well informed by the samples obtained under the previous R_f value. This step-wise procedure enables the search for the area near the global minimum in $A(\mathbf{X})$ to be more and more precise as R_f becomes larger. Eventually, at $R_f \to \infty$, the deterministic version of the model is imposed, and the sampling results become final.

For numerical results, we first tested the PAHMC method on the Lorenz96 model with dimension $D = 20$. We reported on results where the number of observations ranges from $L = 7$ to $L = 12$. We have shown that, at the smallest value of L considered, the sampling of $\mathbf{X}$ provided by PAHMC do not necessarily agree with the observations, though they can be in accordance with the dynamical model. As the number of observations

increases, the method took advantage of the additional information available and successfully located the area near the global minimum of the action. The prediction results also showed that the successful transfer of information from data to the model had enabled predictability of the model system. This success is based on (1) improvements in the estimation of the full state of the model at the termination of the observation window and (2) the stabilization of intrinsic instabilities in the chaotic nonlinear model [79]. A direct comparison between PAHMC and the Random-proposal implementation of Precision Annealing on Lorenz96 showed that PAHMC is approximately 25 times faster in achieving a similar level of mixing.

For the second class of numerical results, we presented the application of PAHMC on a biophysical problem involving the Hodgkin-Huxley model of neuron. This model has $D = 4$, and only the cross-membrane voltage representing $L = 1$ can be observed. Another difference from the Lorenz96 system is that the Hodgkin-Huxley model is a representative example of a *driven* system. This poses an additional challenge. We showed that PAHMC is still perfectly capable of handling the transfer of information in this case. The success was, again, measured by the quality of the prediction and parameter estimation.

Aside from the speed race mentioned above, the core of PAHMC can be performed in parallel using GPUs [44, 89, 90]. Contemporary GPU capability could allow for a significant speed-up in HMC calculations. Using PAHMC on very high dimensional tasks of information transfer seems quite promising.

The next chapter explores applying Precision Annealing-like ideas in training classical artificial neural networks.

This dissertation contains material as it appears in Zheng Fang, Adrian S. Wong, Kangbo Hao, Alexander J. A. Ty, and Henry D. I. Abarbanel: Precision annealing Monte Carlo methods for statistical data assimilation and machine learning, *Physical Review Research*, 2(1), 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 5

A Novel Approach for Training Artificial Neural Networks

In the previous chapters, we formulated the data assimilation problem as evaluating an expected-value integral in the form of Eq. (2.11) with the action defined by Eq. (2.13). We also presented extensive examples in which the desired transfer of information from data to physical systems was accomplished.

This chapter explores an entirely new territory: training artificial neural networks (or "deep learning" models) using novel, non-backpropagation approaches that involve Precision Annealing. This idea is inspired by the strong equivalence between an artificial neural network and a dynamical system described by a set of ODEs [15], as we have seen in Chapter 1.

Under the above-mentioned equivalence, we first give a general form of the action in the context of artificial neural networks. Just like the actions for physical models, the structure of the proposed action for deep learning clearly has the model errors incorporated, and it is therefore conceptually different from traditional loss functions [8, 12] which requires backpropagation [91] to evaluate its gradients.

Once the action is presented, one can immediately recognize that training artificial neural networks essentially becomes the familiar task of evaluating certain expected-value integrals as in Eq. (2.11). Indeed, the novel approach proposed in this chapter is exactly about evaluating $\langle G(\mathbf{X})\rangle$ using Precision Annealing combined with HMC or the Laplace's (gradient-descent) method, given the action to be discussed in Sec. 5.1.

One way to demonstrate the efficacy of this novel approach is to use it to train artificial neural networks that cannot be trained by backpropagation. In this spirit, we focus on training in Sec. 5.2. The numerical results clearly indicate that the novel approach is capable of training deep artificial neural networks in the absence of all the "engineering patches" needed by backpropagation.

As a side note, this chapter focuses exclusively on the novel approach and its applications, and we do not review the state-of-the-art models or methods of deep learning. Nonetheless, some working knowledge on contemporary machine learning is expected; interested readers could consult Refs. [7, 8, 9, 10, 11, 12] as needed.

The chapter concludes with a discussion of future works.

5.1 Motivation and Proposal

Recent years have witnessed a surge in both computational capacity and the amount of data produced, which are the two key factors behind the success of deep learning [14]. Artificial neural network, the basic building block of deep learning, is believed to perform better in complicated, real-world tasks when the depth is large [92, 93]. For example, modern natural language processing models usually exceed ten layers in depth and have millions of free parameters [94].

However, training deep artificial neural networks poses significant challenges primarily due to the following two factors.

(1) The landscape of the loss function of a typical deep network is nonconvex and lacks smoothness [95].

(2) The intrinsic instabilities of backpropagation can often cause the so-called vanishing and exploding gradient problems [96, 97].

Recipes such as long short-term memory [13] and ResNet [98] have been proposed to circumvent the above problems. Although working well in some cases, such attempts do not solve the fundamental problems in the construction of loss functions and the use of backpropagation to train deep networks [99], as described in the above list.

In this dissertation, we propose an approach for training artificial neural networks that is conceptually different from the state-of-the-art methods. The approach is inspired by the layer-time correspondence first discussed in Ref. [15], and it shares the same theoretical formulation with PAHMC.

5.1.1 Notation

We begin with a survey of notations that largely resemble those established for data assimilation tasks. Within an M-layer, feed-forward artificial neural network, m denotes the layer index ranging from 1 to M; for a particular layer m, the number of nodes is denoted by D_m.

We use S to denote the number of samples (input-output pairs) for training the network. For the s-th sample in the training data set, $\mathbf{y}^s(1) \in \mathbb{R}^{D_1}$ is the teaching signal for the input layer, and $\mathbf{y}^s(M) \in \mathbb{R}^{D_M}$ is the teaching signal for the output layer.

The state variables of the m-th layer for the s-th sample are denoted by $\mathbf{x}^s(m)$ where we have $\mathbf{x}^s(m) \in \mathbb{R}^{D_m}$. The inter-layer parameters (weights) between layers m and $m+1$ are denoted by $W(m)$ where we have $W(m) \in \mathbb{R}^{D_{m+1} \times (D_m+1)}$. We call the set that concatenates all these variables together a *path* and use $\mathbf{X}$ to denote it. As a result, the

path has the form

$$\mathbf{X} = \{\mathbf{x}^s(m) \,|\, 1 \leq m \leq M,\, 1 \leq s \leq S\} \cup \{W(m) \,|\, 1 \leq m \leq M-1\}\,. \tag{5.1}$$

In addition, $f_i(\mathbf{x}^s(m), W(m); m) \in \mathbb{R}$ is the activation function that takes layer m as the input and outputs on the i-th node of layer $m+1$. Note that we do not require $f_i(\mathbf{x}^s(m), W(m); m)$ to be equal to $x_i^s(m+1)$ as they are in backprop-based machine learning.

The objective function, which we call the *action*, is denoted by $A(\mathbf{X})$. Its form will be given in the following section.

Context-specific notations will be introduced where they first appear.

5.1.2 Action

Here, we present the action as a key equation in our approach. For brevity, we shall write $f_i(\mathbf{x}^s(m), W(m); m)$ as $f_i^s(m)$ except in Eq. (5.3). This notation should not cause ambiguity within the scope of our discussion.

To provide context for a general discussion, and without loss of generality, we focus on feed-forward networks with three or more layers ($M \geq 3$) for classification tasks. The most general form of the action then reads

$$\begin{aligned} A(\mathbf{X}) = {} & \frac{R_m}{2S} \sum_{s=1}^{S} \sum_{a=1}^{D_1} \mathcal{H}_a^s(1) \\ & + \frac{R_m}{2S} \sum_{s=1}^{S} \sum_{a=1}^{D_M} \mathcal{H}_a^s(M) \\ & + \frac{R_f}{2S} \sum_{s=1}^{S} \sum_{m=1}^{M-1} \sum_{a=1}^{D_{m+1}} \mathcal{L}_a^s(m+1 \,|\, m)\,. \end{aligned} \tag{5.2}$$

In Eq. (5.2), $\mathcal{H}_a^s(m)$ is the measurement error at the a-th node of layer m for the s-th

sample, $\mathcal{L}_a^s(m+1\,|\,m)$ is the model error at the a-th node of layer $m+1$ for the s-th sample, and R_m and R_f are scalars. Note that $\mathcal{H}_a^s(m) = 0$ holds for $2 \leq m \leq M-1$ in the above equation, reflecting the fact that there is no observation on the hidden layers.

Fig. 5.1 provides a visualization of the core ideas behind the formulation of our approach. In a traditional loss function, there is no model error, which means that the activation of layer m, i.e., $\mathbf{f}^s(m)$, is always equal to the state of the next layer, $\mathbf{x}^s(m+1)$. However, as shown clearly in the figure and in Eq. (5.2), our interpretation of a network allows room for finite model errors.

Once the general form of the action is established, there are several ways one can specify $\mathcal{H}$ and $\mathcal{L}$ depending on the nature of the training data set. In this dissertation, we work with three mild assumptions that will collectively yield a specific form of the action. They are as follows.

(1) The noise is Gaussian for the input teaching signals.

(2) The teaching signals for the output layer (i.e., the teaching labels) are all correct, which implies $x_a^s(M) = y_a^s(M)$ and $\mathcal{H}_a^s(M) = 0$.

(3) The model error is Gaussian between each pair of adjacent layers.

Consequently, the action now becomes

$$\begin{aligned} A(\mathbf{X}) = & \frac{R_m}{2S} \sum_{s=1}^{S} \sum_{a=1}^{D_1} \left(x_a^s(1) - y_a^s(1)\right)^2 \\ & + \frac{R_f}{2S} \sum_{s=1}^{S} \sum_{m=1}^{M-1} \sum_{a=1}^{D_{m+1}} \left[x_a^s(m+1) - f_a(\mathbf{x}^s(m), W(m); m)\right]^2 . \end{aligned} \tag{5.3}$$

Before we proceed, recall that $x_a^s(M)$ in Eq. (5.3) is a constant under the second assumption above.

We now write down the derivatives of the action as a contrast to what is in the backpropagation procedure.

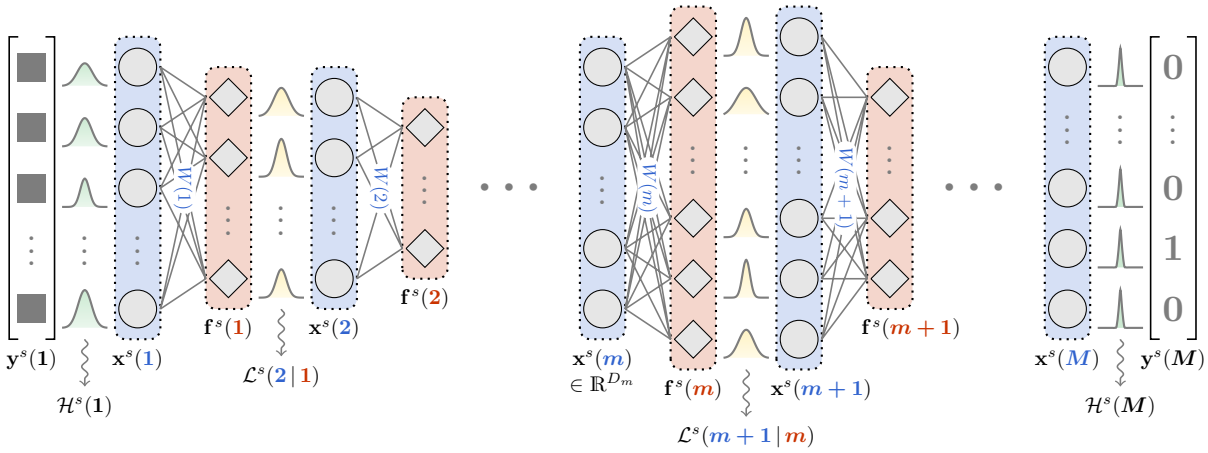

Figure 5.1: A schematic of the structure of a feed-forward network in the presence of model error. The layer index increases from left to right. Each layer is a combination of a blue region and a red region. At the input layer, $\mathbf{y}^s(1)$ denotes a single input sample, and $\mathcal{H}^s(1)$ denotes the corresponding measurement error. For each layer, an activation function parameterized by W takes in state variable $\mathbf{x}^s(m)$ and produces $\mathbf{f}^s(m)$, which is then paired with the state of the next layer, $\mathbf{x}^s(m+1)$, to produce a finite model error. At the output layer, $\mathbf{x}^s(M)$ is paired with data $\mathbf{y}^s(M)$ to produce a measurement error $\mathcal{H}^s(M)$.

If we define $\eta_j^s(m)$ as

$$\eta_j^s(m) = \sum_{i=1}^{D_{m+1}} \left(x_i^s(m+1) - f_i^s(m)\right) \cdot \frac{\partial f_i^s(m)}{\partial x_j^s(m)}, \tag{5.4}$$

for $1 \le s \le S$, $1 \le m \le M-1$, and $1 \le j \le D_m$, the derivative with respect to $x_j^s(m)$ then takes a convenient form

$$\frac{\partial A(\mathbf{X})}{\partial x_j^s(m)} = \begin{cases} \dfrac{R_m}{S}\left(x_j^s(1) - y_j^s(1)\right) - \dfrac{R_f}{S}\eta_j^s(1), & \text{if } m = 1, \\ \dfrac{R_f}{S}\left(x_j^s(m) - f_j^s(m-1) - \eta_j^s(m)\right), & \text{if } 2 \le m \le M-1. \end{cases} \tag{5.5}$$

Similarly, we define $\varphi_{lj}(m)$ as

$$\varphi_{lj}(m) = \sum_{s=1}^{S} \sum_{i=1}^{D_{m+1}} \left(x_i^s(m+1) - f_i^s(m)\right) \cdot \frac{\partial f_i^s(m)}{\partial W_{lj}(m)}, \tag{5.6}$$

for $1 \le m \le M-1$, $1 \le l \le D_{m+1}$, and $1 \le j \le D_m + 1$. Then, the derivatives with respect to the inter-layer parameters, $W_{lj}(m)$, have the form

$$\frac{\partial A(\mathbf{X})}{\partial W_{lj}(m)} = -\frac{R_f}{S}\varphi_{lj}(m). \tag{5.7}$$

As we have mentioned, Eqs. (5.4)–(5.7) differ from backpropagation in a fundamental manner.

Next, we will look at some example activation functions that give rise to specific forms of η and φ.

One remark on the action is that, often in classification tasks, one can choose cross-entropy (a shifted Kullback–Leibler divergence) as the model error for the output layer. Under this choice, there is $\mathcal{L}_a^s(M \,|\, M-1) = -x_a^s(M) \cdot \log f_a^s(M-1)$, which is a slight deviation from the third assumption above.

5.1.3 Example Activation Functions

Before presenting numerical results, we examine the local dynamics of a network, i.e., activation functions. In particular, we discuss two common activation functions and present their derivatives with respect to x and W. For further brevity, we do not explicitly write the dependence on m for x, W, and f.

The first example is rectified linear unit (ReLU). The activation function reads

$$\begin{aligned} z_i^s &= \sum_{j=1}^{D_m} W_{ij}\, x_j^s + W_{i,D_m+1}, \\ f_i^s &= \max\left\{0,\, z_i^s\right\}, \end{aligned} \tag{5.8}$$

for $1 \le s \le S$ and $1 \le i \le D_{m+1}$.

Under the above, Eq. (5.4) becomes

$$\eta_j^s(m) = \sum_{i=1}^{D_{m+1}} \left(x_i^s(m+1) - f_i^s\right) \cdot \mathbb{1}\left\{f_i^s > 0\right\} \cdot W_{ij}\,, \tag{5.9}$$

for $1 \le s \le S$ and $1 \le j \le D_m$.

Eq. (5.6) becomes, for $1 \le l \le D_{m+1}$,

$$\varphi_{lj}(m) = \begin{cases} \displaystyle\sum_{s=1}^{S} \left(x_l^s(m+1) - f_l^s\right) \cdot \mathbb{1}\left\{f_l^s > 0\right\} \cdot x_j^s\,, & \text{if } 1 \le j \le D_m, \\ \displaystyle\sum_{s=1}^{S} \left(x_l^s(m+1) - f_l^s\right) \cdot \mathbb{1}\left\{f_l^s > 0\right\}, & \text{if } j = D_m + 1. \end{cases} \tag{5.10}$$

The second example is Softmax. The activation function reads

$$\begin{aligned} z_i^s &= \sum_{j=1}^{D_m} W_{ij}\, x_j^s + W_{i,D_m+1}, \\ f_i^s &= \frac{\exp\left(z_i^s - \xi^s\right)}{\sum_{i'} \exp\left(z_{i'}^s - \xi^s\right)}, \end{aligned} \tag{5.11}$$

for $1 \le s \le S$ and $1 \le i \le D_{m+1}$. In Eq. (5.11), ξ^s is a constant for overflow protection, which can be set to $\xi^s = \max\{0,\, \max_i \{z_i^s\} - 100\}$ or something similar.

Before we proceed, observe that

$$\frac{\partial f_i^s}{\partial z_k^s} = f_i^s \cdot (\delta_{ik} - f_k^s) \tag{5.12}$$

holds for $1 \le i, k \le D_{m+1}$.

Under the above, Eq. (5.4) becomes

$$\eta_j^s(m) = \sum_{i=1}^{D_{m+1}} (x_i^s(m+1) - f_i^s) \cdot f_i^s \cdot \left(W_{ij} - \sum_{k=1}^{D_{m+1}} f_k^s \, W_{kj} \right), \tag{5.13}$$

for $1 \le s \le S$ and $1 \le j \le D_m$.

Eq. (5.6) becomes, for $1 \le l \le D_{m+1}$,

$$\varphi_{lj}(m) = \begin{cases} \displaystyle\sum_{s=1}^{S} \left[x_l^s(m+1) - f_l^s + \sum_{i=1}^{D_{m+1}} f_i^s \cdot (f_i^s - x_i^s(m+1)) \right] \cdot f_l^s \, x_j^s \,, & \text{if } 1 \le j \le D_m, \\ \displaystyle\sum_{s=1}^{S} \left[x_l^s(m+1) - f_l^s + \sum_{i=1}^{D_{m+1}} f_i^s \cdot (f_i^s - x_i^s(m+1)) \right] \cdot f_l^s \,, & \text{if } j = D_m + 1. \end{cases} \tag{5.14}$$

5.2 Numerical Results

To illustrate the point that the new approach is effective and promising, we present in this section numerical results on training multi-layer, feed-forward networks with Precision Annealing combined with both HMC and the Laplace's method (i.e., pure optimization).

Our focus here is training instead of model design or generalization (prediction). In particular, we train deep networks that have enough parameters to memorize the data sets so that we have a clear idea how well the approach performs.

5.2.1 Precision Annealing with HMC

Let us first look at the results obtained by using the combination of Precision Annealing and HMC.

The model used is a fully-connected, feed-forward artificial neural network with Softmax connections for the output layer and leaky ReLU connections for all other layers. Specifically, the network has $M = 10$ layers. The input layer has dimension $D_1 = 10$, the output layer has dimension $D_M = 10$, and all the eight hidden layers have dimension $D_2 = \cdots = D_9 = 5$.

The training data set is prepared as follows. We generate $S = 10$ input samples $\mathbf{y}^1(1), \cdots, \mathbf{y}^S(1)$, each being a D_1-dimensional random vector, from a joint uniform distribution with range $(-\sqrt{3}, \sqrt{3})$. Then, we generate $S = 10$ output samples (i.e., training labels) $\mathbf{y}^1(M), \cdots, \mathbf{y}^S(M)$, each being a D_M-dimensional vector with one randomly selected component being 1 and all others being 0.

Before training begins, we need to initialize the path $\mathbf{X}$, which consists of the state variables and the inter-layer parameters (weights). For the results presented here, the state variables $\{\mathbf{x}^s(m)\}$ are initialized to be the training data for the input and output layers; for the hidden layers, they are drawn from a normal distribution with $\mu = 0$ and $\sigma^2 = 0.75$. The parameters $\{W(m)\}$ are drawn from a uniform distribution with range $(-1/\sqrt{D_m}, 1/\sqrt{D_m})$.

Although the data are generated randomly, the task is still to "classify" the input data so that the results match the training labels. Note that the network is big enough to be able to memorize these random samples, so any failure will be due to the method used instead of the choice of the model.

Backpropagation failed to train the network on the data set described above. The classification accuracy remained at 10%, which is a result of random guessing.

In contrast, Precision Annealing (combined with HMC) succeeded in the task.

Upon initialization, the action, as described in Eq. (5.3), was at the order of 10 and the classification accuracy was 10%. Then, the action rapidly decreased and the accuracy increased to 100%, as shown in Table 5.1.

Table 5.1: Classification accuracy for three values of R_f.

β	R_f	Number of HMC Samples	$A(\mathbf{X})$	Classification Accuracy (%)
0	1.0	500	2.48×10^{-1}	20
1	3.0	500	8.66×10^{-3}	70
2	9.0	500	1.99×10^{-3}	100

Here are two remarks on the above result. First, recall that the training data are just pure noise, and the model is very flexible and non-physical, we should not expect the approach to extract the "true dynamics" (we know there is none) from the data. Consequently, all action levels reported in Table 5.1 are almost entirely made of *model error*, which means there is no dynamics to be learned. Nevertheless, the accuracy levels show that we have achieved our goal, which is to let the model memorize the data. Second, the accuracy levels are measured on the training data. We do not expect the model trained on pure noise to predict well on some new noise.

5.2.2 Precision Annealing with Laplace's Method

We now turn to the results from combining Precision Annealing with pure optimization of the action (i.e., Laplace's method). We use two data sets here, one is the random data set introduced in Sec. 5.2.1, and the other is the popular MNIST database of handwritten digits [100].

We first report results on the random data. The model is, again, a fully-connected, feed-forward artificial neural network. The activation function for all the layers except the last is leaky ReLU, and the activation for the last layer is Softmax. There are $S = 10$ input-output pairs used in the training. The initialization of the state variables and the inter-layer parameters are the same as in the previous example.

However, the network now has $M = 30$ layers, and the dimension of the input signal is 200. Specifically, the input layer has dimension $D_1 = 200$, the output layer has dimension $D_M = 10$, and all the 28 hidden layers have dimension $D_2 = \ldots = D_{29} = 5$. The depth of this model rules out entirely the possibility of a successful training using backpropagation. Therefore, if we can train this network using the combination of Precision Annealing and Laplace's method, it would be a strong indication that our approach is effective in cases that cannot be handled by backpropagation.

Fig. 5.2 shows the results of the training. On the top panel is the value of the action as the model constraints are gradually enforced. At the beginning, the hyperparameter R_f (defined in Sec. 2.4) is set to be $R_{f_0} = 1.0$. Then, following the Precision Annealing procedure, we have $R_f = R_{f_0} \times \alpha^\beta$ where $\alpha = 2.0$ and $\beta \in \{0, \ldots, 7\}$. For the first stage in which $\beta = 0$, we perform gradient descent on the action for 1000 steps. For all the subsequent stages, we perform 2000 steps for each β. The action clearly follows a decreasing pattern, with expected small jumps at each transitioning point for β. This indicate that information has been transferred from the data set to our model. The bottom panel shows the classification accuracy as a function of step, which is a more direct measure of success.

Next, we report results on the MNIST database of handwritten digits, which is widely used as a test bed for many deep learning methods. Each sample on the input side is a grey-scale image of 28×28 pixels. Each output label is a 10-dimensional vector.

The training set consists of $S = 100$ such input-output pairs. The model used is still a feed-forward network, this time with $M = 20$ layers. The dimension of the input layer is $D_1 = 784$. The first three hidden layers have dimension $D_2 = D_3 = D_4 = 100$, the rest of the hidden layers have dimension $D_5 = \ldots = D_{19} = 20$, and the output layer has dimension $D_M = 10$. The activation functions for all layers except the last is leaky ReLU, and the activation for the output layer is Softmax.

Fig. 5.3 shows the action, loss, and classification accuracy as we gradually enforce

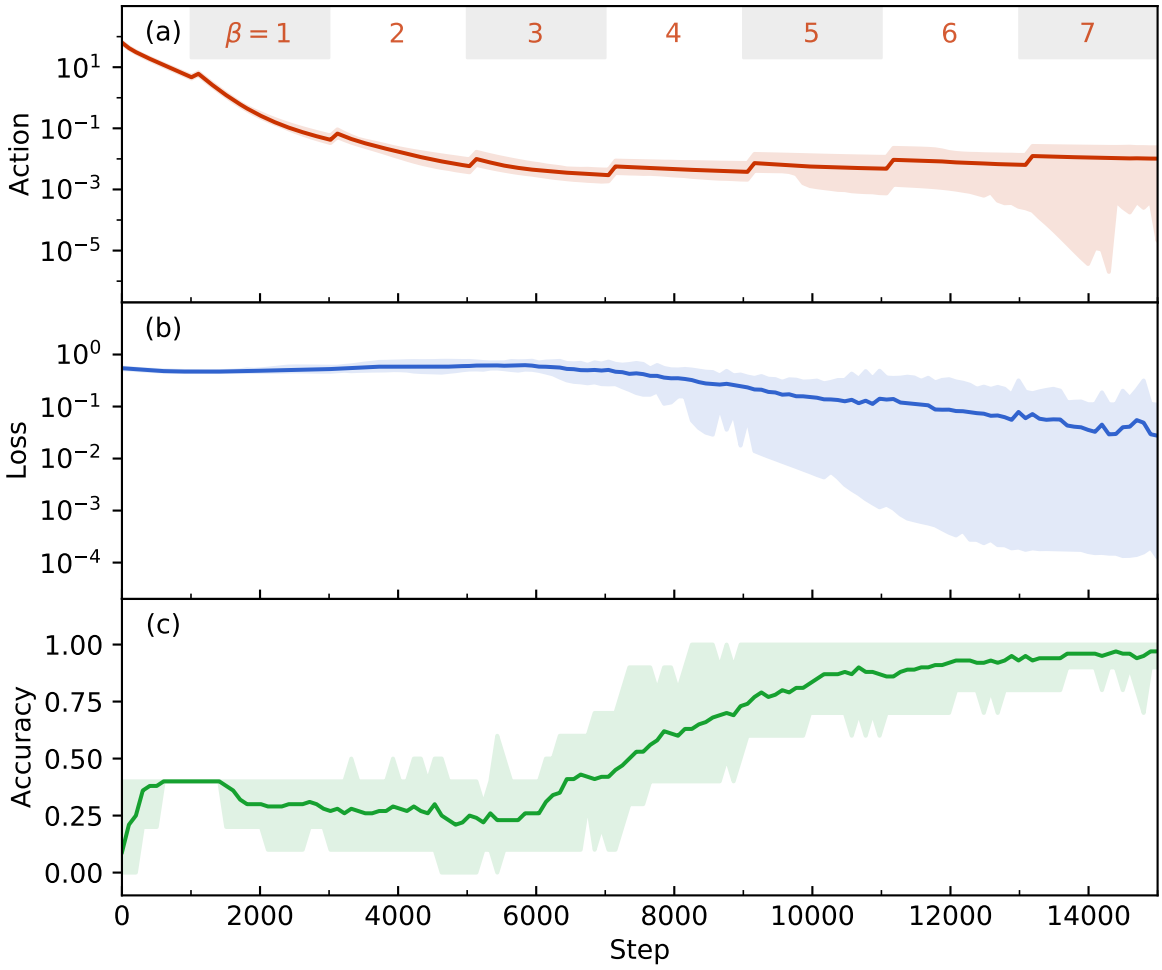

Figure 5.2: Action, loss, and classification accuracy regarding the randomly generated data set. The horizontal axis represents both steps within each R_f and different R_f values. (a) Action as a function of optimization step. The small but sudden changes in the action are due to the increase of β and thus R_f. The error band is shown. (b) Value of the loss function as a function of optimization step. The error band is shown. (c) Classification accuracy on the training set as a function of optimization step. The error band is shown.

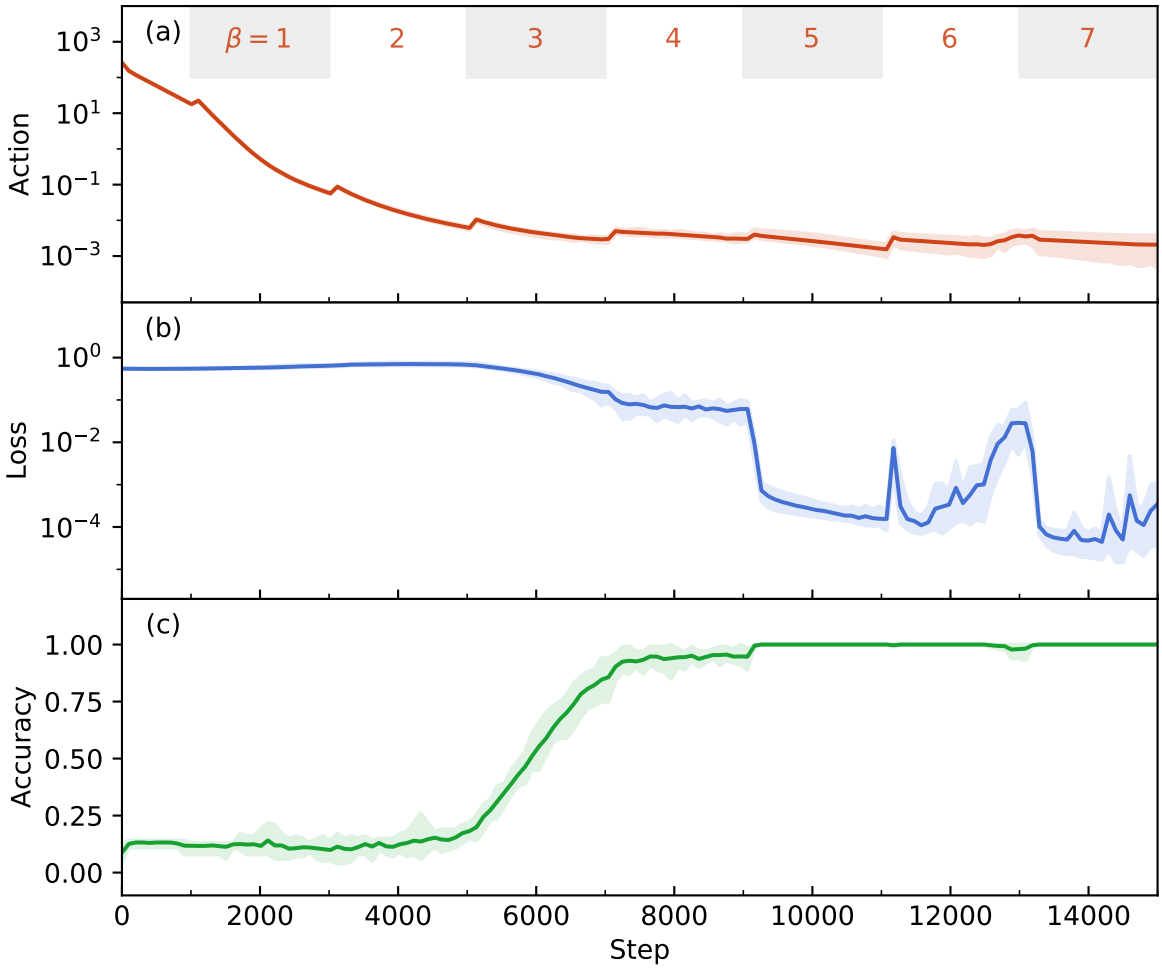

Figure 5.3: Action, loss, and classification accuracy regarding the MNIST database of handwritten digits. The horizontal axis represents both steps within each R_f and different R_f values. The boundaries of the error bands represent the maximum and minimum values among the ten calculations, respectively. (a) Action as a function of optimization step. The small but sudden changes in the action are due to the increase of β and thus R_f. (b) Value of the loss function as a function of optimization step. (c) Classification accuracy on the 100 training images as a function of optimization step.

the model constraints using Precision Annealing. The schedule for the hyperparameter R_f and the number of gradient-descent steps are the same as the ones presented above. The action, as shown in the top panel, decreases from greater than 10^2 to around 10^{-3}. The value of the MSE loss decreases from the order of 1 to below 10^{-3}. The classification accuracy increases from a level of random guessing (around 10%) to 100%.

The above results strongly indicate that our approach is capable of training a broad spectrum of artificial neural networks without a need for backpropagation.

5.3 Future Work

The formulation presented in Sec. 5.1 is novel and theoretically appealing, and the numerical results demonstrated in Sec. 5.2 are promising. Nevertheless, much needs to be done to turn the proposed approach into a full-fledged method capable of training massive deep learning models for data-intensive tasks.

For example, one can use the proposed approach to train an even deeper and wider model on larger data sets (there are many available). Achieving this would better showcase the advantages of the novel approach over backpropagation.

It is also important to obtain results on a variety of other tasks including image classification and time-series analysis. These would involve popular models such as convolutional neural networks and recurrent neural networks.

Moreover, the topic of prediction needs to be thoroughly studied. Do models trained by the proposed approach generalize better on test data than those trained by backpropagation? Which activation functions are preferred? Is regularization as effective as it is in low-dimensional models? There are many questions to be addressed.

And there is much work to do.

Bibliography

[1] Henry D. I. Abarbanel. *Predicting the future: completing models of observed complex systems*. Understanding Complex Systems. Springer-Verlag New York, 2013.

[2] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Earth Sciences & Geography. Springer-Verlag Berlin Heidelberg, 2009.

[3] Henning U. Voss, Jens Timmer, and Jürgen Kurths. Nonlinear dynamical system identification from uncertain and indirect measurements. *International Journal of Bifurcation and Chaos*, 14(6):1905–1933, 2004.

[4] Ulrich Parlitz. Estimating model parameters from time series by autosynchronization. *Physical Review Letters*, 76(8):1232, 1996.

[5] Paul So, Edward Ott, and W. P. Dayawansa. Observing chaos: deducing and tracking the state of a chaotic system from limited observation. *Physical Review E*, 49(4):2650, 1994.

[6] T. N. Krishnamurti. Numerical weather prediction. *Annual Review of Fluid Mechanics*, 27:195–225, 1995.

[7] Li Deng and Dong Yu. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3-4):197–387, 2014.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive Computation and Machine Learning. The MIT Press, 2016.

[9] Jürgen Schmidhuber. Deep learning in neural networks: an overview. *Neural Networks*, 61:85–117, 2015.

[10] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[11] M. I. Jordan and T. M. Mitchell. Machine learning: trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[12] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G. R. Day, Clint Richardson, Charles K. Fisher, and David J.Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, 2019.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[15] Henry D. I. Abarbanel, Paul J. Rozdeba, and Sasha Shirman. Machine learning: deepest learning as statistical data assimilation problems. *Neural Computation*, 30(8):2025–2055, 2018.

[16] Zheng Fang, Adrian S. Wong, Kangbo Hao, Alexander J. A. Ty, and Henry D. I. Abarbanel. Precision annealing Monte Carlo methods for statistical data assimilation and machine learning. *Physical Review Research*, 2(1):013050, 2020.

[17] Alexander J. A. Ty, Zheng Fang, Rivver A. Gonzalez, Paul. J. Rozdeba, and Henry D. I. Abarbanel. Machine learning of time series using time-delay embedding and precision annealing. *Neural Computation*, 31(10):2004–2024, 2019.

[18] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

[19] Barry Saltzman. Finite amplitude free convection as an initial value problem—I. *Journal of the Atmospheric Sciences*, 19(4):329–341, 1962.

[20] Henry D. I. Abarbanel. *Analysis of observed chaotic data*. Institute for Nonlinear Science. Springer-Verlag New York, 1996.

[21] Alan L. Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[22] Daniel Johnston and Samuel Miao-Sin Wu. *Foundations of cellular neurophysiology*. The MIT Press, 1994.

[23] David Sterratt, Bruce Graham, Andrew Gillies, and David Willshaw. *Principles of computational modelling in neuroscience*. Cambridge University Press, 2011.

[24] Alain Nogaret, C. Daniel Meliza, Daniel Margoliash, and Henry D. I. Abarbanel. Automatic construction of predictive neuron models through large scale assimilation of electrophysiological data. *Scientific Reports*, 6:32749, 2016.

[25] Joseph Pedlosky. *Geophysical fluid dynamics*. Springer-Verlag New York, 1987.

[26] Geoffrey K. Vallis. *Atmospheric and oceanic fluid dynamics: fundamentals and large-scale circulation*. Cambridge University Press, 2017.

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

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

[29] Richard A. Anthes. Data assimilation and initialization of hurricane prediction models. *Journal of Atmospheric Sciences*, 3:702–719, 1974.

[30] Henry D. I. Abarbanel, Daniel R. Creveling, Reza Farsian, , and Mark Kostuk. Dynamical state and parameter estimation. *SIAM Journal on Applied Dynamical Systems*, 8(4):1341–1381, 2009.

[31] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Texts in Statistics. Springer-Verlag New York, 2009.

[32] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning: with applications in R*. Springer Texts in Statistics. Springer-Verlag New York, 2013.

[33] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes: the art of scientific computing*. Cambridge University Press, 2007.

[34] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

[35] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[36] Michael Betancourt. The convergence of Markov chain Monte Carlo methods: from the Metropolis method to Hamiltonian Monte Carlo. *Annalen der Physik*, 531(3):1700214, 2018.

[37] Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letter B*, 195(2):216–222, 1987.

[38] Radford M. Neal. MCMC using Hamiltonian dynamics. In Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, editors, *Handbook of Markov chain Monte Carlo*, chapter 5, pages 113–162. Chapman and Hall/CRC, 2011.

[39] Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *ArXiv e-prints*, 2017.

[40] Robert M. Fano. *Transmission of information: a statistical theory of communication*. The MIT Press, 1961.

[41] Sergios Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic Press, 2020.

[42] Pierre Simon Laplace. Memoir on the probability of the causes of events. *Statistical Science*, 1(3):364–378, 1986. Translation to English by S. M. Stigler.

[43] Mark Kostuk, Bryan A. Toth, C. Daniel Meliza, Daniel Margoliash, and Henry D. I. Abarbanel. Dynamical estimation of neuron and network properties II: path integral Monte Carlo methods. *Biological Cybernetics*, 106(3):155–167, 2012.

[44] John C. Quinn. *A path integral approach to data assimilation in stochastic nonlinear systems*. PhD thesis, University of California San Diego, 2010.

[45] Jingxin Ye. *Systematic annealing approach for statistical data assimilation*. PhD thesis, University of California San Diego, 2016.

[46] Paul J. Rozdeba. *Nonlinear inference in partially observed physical systems and deep neural networks*. PhD thesis, University of California San Diego, 2018.

[47] Jingxin Ye, Nirag Kadakia, Paul J. Rozdeba, Henry D. I. Abarbanel, and John C. Quinn. Improved variational methods in statistical data assimilation. *Nonlinear Processes in Geophysics*, 22:205–213, 2015.

[48] Jingxin Ye, Daniel Rey, Nirag Kadakia, Michael Eldridge, Uriel I. Morone, Paul Rozdeba, Henry D. I. Abarbanel, and John C. Quinn. Systematic variational method for statistical nonlinear state and parameter estimation. *Physical Review E*, 92(5):052901, 2015.

[49] Jean Zinn-Justin. *Quantum field theory and critical phenomena.* International Series of Monographs on Physics. Clarendon Press, 2002.

[50] Gareth O. Roberts and Jeffrey S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–714, 2004.

[51] J. R. Norris. *Markov chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.

[52] Krishna B. Athreya, Hani Doss, and Jayaram Sethuraman. On the convergence of the Markov chain simulation method. *The Annals of Statistics*, 24(1):69–100, 1996.

[53] Luke Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

[54] Richard E. Bellman. *Dynamic programming*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 2010.

[55] Richard E. Bellman. *Adaptive control processes: a guided tour*. Princeton Legacy Library. Princeton University Press, 2015.

[56] Vasilios I. Manousiouthakis and Michael W. Deem. Strict detailed balance is unnecessary in Monte Carlo simulation. *The Journal of Chemical Physics*, 110(6):2753–2756, 1999.

[57] Persi Diaconis, Susan Holmes, and Radford M. Neal. Analysis of a nonreversible Markov chain sampler. *The Annals of Applied Probability*, 10(3):726–752, 2000.

[58] Radford M. Neal. Improving asymptotic variance of MCMC estimators: non-reversible chains are better. *ArXiv e-prints*, 2004.

[59] Herbert Goldstein, Charles P. Poole, and John L. Safko. *Classical mechanics*. Pearson, 2001.

[60] V. I. Arnold. *Mathematical methods of classical mechanics*. Graduate Texts in Mathematics. Springer-Verlag New York, 1989.

[61] Nilesh Tripuraneni, Mark Rowland, Zoubin Ghahramani, and Richard Turner. Magnetic Hamiltonian Monte Carlo. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3453–3461. PMLR, 2017.

[62] Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2009.

[63] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*. Springer Series in Computational Mathematics. Springer-Verlag Berlin Heidelberg, 2006.

[64] Nirag Kadakia. Hybrid Monte Carlo with chaotic mixing. *ArXiv e-prints*, 2016.

[65] Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[66] Michael Betancourt. A general metric for Riemannian manifold Hamiltonian Monte Carlo. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, pages 327–334. Springer-Verlag Berlin Heidelberg, 2013.

[67] Ziyu Wang, Shakir Mohamed, and Nando Freitas. Adaptive Hamiltonian and Riemann manifold Monte Carlo samplers. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1462–1470. PMLR, 2013.

[68] Ge Zhong and Jerrold E.Marsden. Lie-Poisson Hamilton-Jacobi theory and Lie-Poisson integrators. *Physics Letters A*, 133(3):134–139, 1988.

[69] Jesús M. Sanz-Serna. Symplectic integrators for Hamiltonian problems: an overview. *Acta Numerica*, 1:243–286, 1992.

[70] Haruo Yoshida. Recent progress in the theory and application of symplectic integrators. In Rudolf Dvorak and Jacques Henrard, editors, *Qualitative and Quantitative Behaviour of Planetary Systems*, pages 27–43. Springer Dordrecht, 1993.

[71] Nawaf Bou-Rabee and Jesús M. Sanz-Serna. Geometric integrators and the Hamiltonian Monte Carlo method. *Acta Numerica*, 27:113–206, 2018.

[72] Sergio Blanes, Fernando Casas, and J. M. Sanz-Serna. Numerical integrators for the Hybrid Monte Carlo method. *SIAM Journal on Scientific Computing*, 36(4):1556–A1580, 2014.

[73] Oren Mangoubi and Nisheeth K. Vishnoi. Dimensionally tight bounds for second-order Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, 2018.

[74] Michael Creutz. Global Monte Carlo algorithms for many-fermion systems. *Physical Review D*, 38(4):1228, 1988.

[75] Sourendu Gupta, A. Irbäc, F. Karsch, and B. Petersson. The acceptance probability in the hybrid Monte Carlo method. *Physics Letters B*, 242(3):437–443, 1990.

[76] Andrew Gelman, Walter R Gilks, and Gareth O. Roberts. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997.

[77] Edward N. Lorenz. Predictability – a problem partly solved. In Tim Palmer and Renate Hagedorn, editors, *Predictability of weather and climate*, chapter 3, pages 40–58. Cambridge University Press, 2006.

[78] Edward N. Lorenz and Kerry A. Emanuel. Optimal sites for supplementary weather observations: simulation with a small model. *Journal of the Atmospheric Sciences*, 55(3):399–414, 1998.

[79] Mark Kostuk. *Synchronization and statistical methods for the data assimilation of HVC neuron models*. PhD thesis, University of California San Diego, 2012.

[80] Owen P. Hamill, A. Marty, E. Neher, B. Sakmann, and Frederick J. Sigworth. Improved patch-clamp techniques for high-resolution current recording from cells and cell-free membrane patches. *Pflügers Archiv*, 391(2):85–100, 1981.

[81] Bryan A. Toth, Mark Kostuk, C. Daniel Meliza, Daniel Margoliash, and Henry D. I. Abarbanel. Dynamical estimation of neuron and network properties I: variational methods. *Biological Cybernetics*, 105(3):217–237, 2011.

[82] Gilles Laurent, Mark Stopfer, Rainer W. Friedrich, Misha I. Rabinovich, Alexander Volkovskii, and Henry D. I. Abarbanel. Odor encoding as an active, dynamical process: experiments, computation, and theory. *Annual Review of Neuroscience*, 24(1):263–297, 2001.

[83] Daniel Johnston and Samuel Miao-Sin Wu. *Foundations of cellular neurophysiology*. MIT Press, 1994.

[84] Christof Koch. *Biophysics of computation: information processing in single neurons*. Computational Neuroscience Series. Oxford University Press, 2004.

[85] Chris Knowlton, C. Daniel Meliza, Daniel Margoliash, and Henry D. I. Abarbanel. Dynamical estimation of neuron and network properties III: network analysis using neuron spike times. *Biological Cybernetics*, 108(3):261–273, 2014.

[86] C. Daniel Meliza, Mark Kostuk, Hao Huang, Alain Nogaret, Daniel Margoliash, and Henry D. I. Abarbanel. Estimating parameters and predicting membrane voltages with conductance-based neuron models. *Biological Cybernetics*, 108(4):495–516, 2014.

[87] Andrew C. Lorenc. Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 112(474):1177–1194, 1986.

[88] Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

[89] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1–19, 1995.

[90] Sergei Prokhorenko, Kruz Kalke, Yousra Nahas, and Laurent Bellaiche. Large scale hybrid Monte Carlo simulations for structure and property prediction. *npj Computational Materials*, 4(1):1–7, 2018.

[91] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[93] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[94] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 4171–4186. Association for Computational Linguistics, 2019.

[95] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[96] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[97] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1310–1318. PMLR, 2013.

[98] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[99] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256. PMLR, 2010.

[100] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324. IEEE, 1998.