ABSTRACT

Title of dissertation: MACHINE LEARNING APPROACHES FOR DATA-DRIVEN ANALYSIS AND FORECASTING OF HIGH-DIMENSIONAL CHAOTIC DYNAMICAL SYSTEMS

Jaideep Pathak
Doctor of Philosophy, 2019

Dissertation directed by: Professor Edward Ott
Department of Physics

We consider problems in the forecasting of large, complex, spatiotemporal chaotic systems and the possibility that machine learning might be a useful tool for significant improvement of such forecasts. Focusing on weather forecasting as perhaps the most important example of such systems, we note that physics-based weather models have substantial error due to various factors including imperfect modeling of subgrid-scale dynamics and incomplete knowledge of physical processes. In this thesis, we ask if machine learning can potentially correct for such knowledge deficits.

First, we demonstrate the effectiveness of using machine learning for model-free prediction of spatiotemporally chaotic systems of arbitrarily large spatial extent and attractor dimension purely from observations of the system's past evolution. We present a parallel scheme with an example implementation based on the reservoir computing paradigm and demonstrate the scalability of our scheme using the

Kuramoto-Sivashinsky equation as an example of a spatiotemporally chaotic system.

We then demonstrate the use of machine learning for inferring fundamental properties of dynamical systems, namely the Lyapunov exponents, purely from observed data. We obtain results of unprecedented fidelity with our novel technique, making it possible to find the Lyapunov exponents of large systems where previously known techniques have failed.

Next, we propose a general method that combines a physics-informed knowledge-based model and a machine learning technique to build a hybrid forecasting scheme. We further extend our hybrid forecasting approach to the difficult case where only partial measurements of the state of the dynamical system are available. For this purpose, we propose a novel technique that combines machine learning with a data assimilation method called an Ensemble Transform Kalman Filter (ETKF).

MACHINE LEARNING APPROACHES FOR DATA-DRIVEN ANALYSIS AND FORECASTING OF HIGH-DIMENSIONAL CHAOTIC DYNAMICAL SYSTEMS

by

Jaideep Satyajit Pathak

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Dedication

Dedicated to my mother, who first taught me math, and my father, who has always given me his unwavering support.

Acknowledgments

I owe a tremendous debt of gratitude to my advisor, Dr. Edward Ott, who has guided and mentored me with a lot of kindness and patience. I have enjoyed working on incredibly interesting and challenging projects under his very able guidance. I consider myself very fortunate to have had the opportunity to work with and learn from such a remarkable person.

My co-advisor, Dr. Michelle Girvan has always made sure she was available for scientific and professional advice when I have needed it despite her many responsibilities. Her scientific acumen and work ethic is inspirational and I thank her for guiding my research.

I would also like to thank Dr. Brian Hunt. Without his helpful insights, brilliant ideas, hard work and expertise, this thesis would not have been possible.

Thanks to Dr. Rajarshi Roy, Dr. Bill Dorland, Dr. Tom Antonsen, and Dr. Daniel Lathrop for helpful discussions and feedback during various stages of my research.

I have been very fortunate to have been part of a thriving culture of collaboration within the graduate student body at the University of Maryland. My friends Sarthak Chandra, Sarah Burnett, Zhixin Lu, Alex Wikner and Joe Hart have been unwavering in their help, support and motivation every time I have needed it. Thanks are due to my collaborators Dr. Istvan Szunyogh, Dr. Garrett Katz and Troy Arcomano.

I am very grateful to the team maintaining the Deepthought-2 High-Performance

Computing (HPC) cluster which enabled a large part of my research. I thank the administrative staff at IREAP and the Department of Physics, particularly Josiland Chambers and Jessica Crossby.

I owe my most profound thanks to my mother, Dr. Padmini Pathak and my father, Dr. Satyajit Pathak for their tremendous support through every stage of my life. Thanks to my sister, Dr. Priya Pathak for always being there for me.

I would like to thank my girlfriend, Sasha Mehan, for all her love, support and encouragement. Thanks to Kaustubh, Ani and Arushi whose friendship has truly meant a lot to me through graduate school.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

IREAP	Institute for Research in Electronics and Applied Physics
RC	Reservoir Computing
KS	Kuramoto-Sivashinsky
RMSE	Root Mean Square Error
ML	Machine Learning
LSTM	Long Short-Term Memory
ESN	Echo State Network
KF	Kalman Filter
ETKF	Ensemble Transform Kalman Filter

Chapter 1: Introduction

1.1 Overview

This thesis is motivated by the crucial problem of forecasting chaotic dynamical systems. This problem has a long and rich history, starting with Edward Lorenz's seminal paper titled "Deterministic Nonperiodic Flow" [2]. Lorenz's paper was motivated by an accidental discovery he made when running computer simulations of a weather model. He noted that a small change in the initial conditions of the weather model changed the outcome of the simulation drastically. His discovery gave rise to the science of predictability and the understanding that deterministic systems could be unpredictable on long enough time scales. Lorenz's discovery has had huge implications for science, particularly, weather forecasting. In the decades that followed Lorenz's paper in 1963, a large number of scientists have contributed to the development of the science of nonlinear systems that are deterministic yet unpredictable, including James Yorke (who first referred to such deterministic unpredictable systems as 'chaotic' [3]), Robert May (who studied unpredictable behavior in ecological models [4]), and Stephen Smale [5] among many others. Several pioneering contributions were made in the field of chaotic dynamics by my thesis advisor, Edward Ott, most notable of them being the study of controlling chaotic dynamical systems with his colleagues Grebogi and Yorke [6].

With the understanding that some highly important dynamical systems are chaotic and thus, inherently unpredictable on long enough time scales, it is crucial to develop mathematical models that can push the envelope of predictablilty as far as theoretically possible. We note that the state of the Earth's atmosphere, or the weather, is perhaps the most important examples of a chaotic dynamical system Developing accurate forecasting models of the weather is therefore of crucial importance. Weather forecasts impact the lives of many millions of people, e.g., by providing warnings of destructive events, like hurricanes or snowstorms. Currently used weather forecasting employs physics-based models (the equations of fluid dynamics, radiative heat transfer, etc.), plus geographical knowledge of mountains, oceans, etc. The models in, however, have substantial error, which, for example, may arise due to imperfect modeling of crucial subgrid-scale dynamics (like clouds, turbulent atmospheric motions, and interactions with small-scale geographic features).

Beginning in the late 1900s, the field of machine learning has seen a revolution. Starting, perhaps, from the invention of the perceptron [7] (an early version of a neural network that could recognize and classify patterns) in 1958, machine learning has advanced rapidly and boasts impressive achievements in a number of fields. Machine learning algorithms today have defeated the best human Go players [8], are capable of facial recognition [9] and image classification [10] at unprecedented scale and accuracy, and are a core component of many technological developments.

Due to the impressive results machine learning has achieved in modeling and data analysis, one might be inclined to ask if it might be helpful in understanding and analyzing chaotic dynamical systems and if it would be possible to use machine learning as a tool to push the envelope of predictability of chaotic dynamical systems. One of the most important early results in this field came from Herbert Jaeger and Harald Haass [11]. In this paper, they showed that it was possible to use a particular kind of neural network, called a reservoir computer, to forecast the Mackey-Glass dynamical system, a system very similar in nature to the dynamical system from Lorenz's seminal paper that gave rise to the science of predictability. The key elements of Jaeger and Haass' paper were (1) they could forecast the chaotic system using only knowledge of the past measurements of the state of the system so that no knowledge of the dynamical system or its mathematical description was required; (2) The technique could be easily generalized to other low-dimensional chaotic systems. This machine learning technique was thus better than any previously known equation-free modeling techniques for chaotic dynamical systems. Jaeger and Haass' reservoir computer forecasting technique, as presented, could only forecast low-dimensional chaotic systems. Chaotic systems of interest such as the weather are incredibly high-dimensional and have vastly greater complexity than the simplistic Lorenz or Mackey-Glass equations.

In this thesis, we consider the use of machine learning for forecasting and analyzing large, highly complex spatiotemporal dynamical systems with a view towards developing techniques that could ultimately improve weather forecasts to a significant extent. This thesis is divided into the following chapters, each of which considers a crucial aspect of data-driven forecasting and analysis of chaotic dynamical systems.

1.2 Model-Free Prediction of Large Spatiotemporally Chaotic Dynamical Systems

In Chapter 2, We demonstrate the effectiveness of using machine learning for model-free prediction of spatiotemporally chaotic systems *of arbitrarily large spatial extent and attractor dimension* purely from observations of the system's past evolution. We present a parallel scheme with an example implementation based on the reservoir computing paradigm and demonstrate the scalability of our scheme using the Kuramoto-Sivashinsky equation as an example of a spatiotemporally chaotic system.

1.3 Using Machine Learning for Data-Driven Analysis of Chaotic Dynamical Systems

In Chapter 3, we use recent advances in the machine learning area known as 'reservoir computing' to formulate a method for model-free estimation from data of the Lyapunov exponents of a chaotic process. The technique uses a limited time series of measurements as input to a high-dimensional dynamical system called a 'reservoir'. After the reservoir's response to the data is recorded, linear regression is used to learn a large set of parameters, called the 'output weights'. The learned output weights are then used to form a modified autonomous reservoir designed to be capable of producing *arbitrarily long* time series whose ergodic properties approximate those of the input signal. When successful, we say that the autonomous reservoir reproduces the attractor's 'climate'. Since the reservoir equations and output weights are known, we can compute derivatives needed to determine the Lyapunov exponents of the autonomous reservoir, which we then use as estimates of the Lyapunov exponents for the original input generating system. We illustrate the effectiveness of our technique with two examples, the Lorenz system, and the Kuramoto-Sivashinsky (KS) equation. In particular, we use the Lorenz system to show that achieving climate reproduction may require tuning of the reservoir parameters. For the case of the KS equation, we note that as the system's spatial size is increased, the number of Lyapunov exponents increases, thus yielding a challenging test of our method, which we find the method successfully passes.

1.4 Model-Assisted Prediction of Chaotic Dynamical Systems

A model-based approach to forecasting chaotic dynamical systems utilizes knowledge of the mechanistic processes governing the dynamics to build an approximate mathematical model of the system. In contrast, machine learning techniques have demonstrated promising results for forecasting chaotic systems purely from past time series measurements of system state variables (training data), without prior knowledge of the system dynamics. The motivation for this chapter is the potential of machine learning for filling in the gaps in our underlying mechanistic knowledge that cause widely-used knowledge-based models to be inaccurate. Thus we here propose a general method that leverages the advantages of these two approaches by combining a knowledge-based model and a machine learning technique to build a hybrid forecasting scheme. Potential applications for such an approach are numerous (e.g., improving weather forecasting). We demonstrate and test the utility of this approach using a particular illustrative version of a machine learning known as reservoir computing, and we apply the resulting hybrid forecaster to a low-dimensional chaotic system, as well as to a high-dimensional spatiotemporal chaotic system. These tests yield extremely promising results in that our hybrid technique is able to accurately predict for a much longer period of time than either its machine-learning component or its model-based component alone.

1.5 Reservoir Observers: Model-free Inference of Unmeasured Variables in Chaotic Dynamical Systems

Deducing the state of a dynamical system as a function of time from a limited number of concurrent system measurements is an important problem of great practical utility. A scheme that accomplishes this is called an "observer". We consider the case in which a model of the system is unavailable or insufficiently accurate, but "training" time series data of the desired state variables are available for a short period of time, and a limited number of other system variables are continually measured. We propose a solution to this problem using networks of neuron-like units known as "reservoir computers." The measurements that are continually available are input to the network, which is trained with the limited-time data to output estimates of the desired state variables. We demonstrate our method, which we call a “reservoir observer”, using the spatiotemporally chaotic Kuramoto-Sivashinsky equation. Subject to the condition of observability (i.e., whether it is in principle possible, by any means, to infer the desired unmeasured variables from the measured variables), we show that the reservoir observer can a very effective and versatile tool for robustly reconstructing unmeasured dynamical system variables.

1.6 Reconstruction and Forecasting of Chaotic Dynamical Systems using Partial Measurements, Imperfect Modeling and Machine Learning Assisted Data Assimilation

In Chapter 6, we consider the problem of data-driven forecasting of chaotic dynamical systems when the available data is sparse. Recently there have been several promising data-driven approaches to forecasting of chaotic dynamical systems using machine learning. Particularly promising among these are hybrid approaches that combine machine learning with a knowledge-based model where a machine learning technique is used to correct the imperfections in the knowledge-based model. Such a bybrid approach is promising when a knowledge-based model is available but is imperfect due to incomplete understanding of the physical processes in the underlying dynamical system. However, previously proposed data-driven forecasting approaches assume knowledge of the full state of the dynamical system. We seek to relax this assumption by using an Ensemble-based Kalman Filter along with machine learning in a novel technique that greatly improves forecasting results. We demonstrate this technique using the simple 3-variable Lorenz dynamical system and the

Kuramoto-Sivashinsky system. We demonstrate that using partial measurements of the state of the dynamical system we can train a machine learning model to correct an imperfect knowledge-based model and greatly improve the prediction quality.

Chapter 2: Model-Free Prediction of Large Spatiotemporally Chaotic Systems: A Reservoir Computing Approach

This chapter is based on the paper, Model-Free Prediction of Large Spatiotemporally Chaotic Systems: A Reservoir Computing Approach, by Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu and Edward Ott, Phys. Rev. Lett. 120, 024102 (2018). ©by the American Physical Society.

2.1 Introduction

Recently, machine learning techniques have proven useful for a wide variety of tasks, from speech recognition [12] to playing Go [8]. In this chapter we show that machine learning can be used for model-free prediction of the evolution of the state of a large spatiotemporally chaotic system. The accomplishment of this task is of great potential application, e.g., for prediction of geophysical dynamical systems. Specifically, we consider a situation where a mechanistic description of the dynamics is unavailable or insufficient for the desired purpose, but reasonably accurate and complete observational data for the evolution of the state of the system of interest can be obtained. Assuming this situation, the goal of this chapter is to formulate an effective technique for predicting the future evolution of very large spatiotemporally chaotic systems from data, an especially difficult problem presently without a robust solution using existing techniques. We note that model-free techniques for prediction based on delay coordinate embedding are well established [13]. These techniques are effective for *low dimensional* chaos, and extensions have been proposed for large spatiotemporally chaotic systems [14]. Within the machine learning community, there have been a number of rapid advances in prediction using the technique known as reservoir computing [1, 15, 16]. In particular, Jaeger and Haas [11] have applied reservoir computing to predict low dimensional chaotic systems with good results. Although we focus on reservoir computing, we expect that other machine learning techniques, e.g., deep learning [17, 18], might also be useful for the task we address. On the other hand, we speculate that, because of their essential dynamical character (see below), artificial neural networks with recurrent connections [19–21], such as reservoir computers, may be inherently well-suited for tasks which are themselves dynamical in character such as prediction or inference of unmeasured state variables of a deterministic system [22]. We find that our reservoir-based spatiotemporal prediction technique yields excellent prediction results of unprecedented quality at reasonable expense.

2.2 Reservoir Computing Configuration

We now briefly introduce the basic ideas of reservoir computing. An input vector $\mathbf{u}(t)$ of dimension D_{in} (Fig. 2.1(a)) is coupled through an I/R (input-to-reservoir) coupler to a high dimensional dynamical system (labeled R in Fig. 2.1(a)) called the "reservoir", from which an output vector $\mathbf{v}(t)$ of dimension D_{out} is coupled through an R/O (reservoir-to-output) coupler. The R/O coupler is assumed to depend on many (D_p) adjustable parameters $\mathbf{p}$, and to create outputs $\mathbf{v}(t)$ that depend linearly upon the parameters $\mathbf{p}$. Denoting the state of the D_r dimensional reservoir by the vector $\mathbf{r}(t)$, the I/R, reservoir, and R/O functions can be represented in discrete time $(t = 0, \Delta t, 2\Delta t, \dots)$ by $\mathbf{r}(t+\Delta t) = \mathbf{G}[\mathbf{r}(t), \mathbf{W}_{in}(\mathbf{u}(t))]$, $\mathbf{v}(t) = \mathbf{W}_{out}[\mathbf{r}(t), \mathbf{p}]$, where $\mathbf{W}_{in}$ (respectively $\mathbf{W}_{out}$) is a mapping from the D_{in} (D_r) dimensional input state space (reservoir state space) to the D_r (D_{out}) dimensional reservoir state space (output state space). We note that while, in this chapter, we consider time to be discrete (and will subsequently take Δt to be small), the analogous continuous time reservoir is also commonly employed. The goal is to train this system to make $\mathbf{v}(t)$ closely approximate the desired outputs $\mathbf{v}_d(t)$ appropriate to the inputs $\mathbf{u}(t)$ (e.g., if the function of the system is speech recognition [12], $\mathbf{u}(t)$ might be an electronic signal derived from a person speaking, while $\mathbf{v}_d(t)$ would represent the words being spoken). To accomplish this, one uses training data consisting of pre-recorded and stored measurements of $\mathbf{u}(t)$ and the resulting $\mathbf{r}(t)$ in some time interval, $-T \leq t \leq 0$, and chooses the output parameters $\mathbf{p}$ so as to minimize the least squares difference between $\mathbf{v}_d(t)$ and $\mathbf{v}(t)$ over the time interval $-T \leq t \leq 0$. Since $\mathbf{v} = \mathbf{W}_{out}[\mathbf{r}, \mathbf{p}]$ is assumed to be linear in the parameters $\mathbf{p}$, the problem of determining $\mathbf{p}$, and hence $\mathbf{W}_{out}$, is a simple linear regression [23]. With $\mathbf{p}$ determined, if all goes well, the reservoir system can be used to fulfill its intended task for $t \geq 0$. Indeed, for large enough D_p and D_r, this framework has proven to be extremely successful for a variety of tasks [1].

Here we are interested in the task of predicting the future, $t > 0$, evolution of $\mathbf{u}(t)$ from training data in $-T \leq t \leq 0$. The prediction task via reservoir computing has been previously addressed with excellent results for a situation where $\mathbf{u}(t)$ comes from the state of a low dimensional chaotic system [11]. In that reference, the desired output condition was that $\mathbf{v}(t)$ be a good approximation to $\mathbf{u}(t)$ (i.e., $\mathbf{v}_d(t) = \mathbf{u}(t)$). After "training" $\mathbf{v}(t)$ to approximate $\mathbf{u}(t)$, the future evolution of $\mathbf{u}(t)$ for $t > 0$ is predicted by replacing the input $\mathbf{u}(t)$ in Fig. 2.1(a) by $\mathbf{v}(t)$, as shown in Fig. 2.1(b). As we will demonstrate, prediction with a single reservoir becomes computationally unfeasible as D_{in} increases. We will propose and illustrate a solution to this problem for spatiotemorally chaotic systems using parallel reservoirs assigned to different spatial regions.

In this chapter, we focus on the following specific implementation choices, which are similar to those in Ref. [11]. (We emphasize here that our choices are illustrative and that many others are possible.) The I/R coupler is $\mathbf{W}_{in}(\mathbf{u}) = \mathbf{W}_{in}\mathbf{u}$ (where $\mathbf{W}_{in}$ is a $D_r \times D_{in}$ matrix whose input elements are drawn from a uniform distribution in $[-\sigma, \sigma]$). The reservoir is a large, low degree (κ), directed Erdös-Rényi network with a $D_r \times D_r$ adjacency matrix $\mathbf{A}$, appropriately scaled so that its largest eigenvalue is equal to ρ. The state of each network node j is a scalar $r_j(t)$ which, in the set-up of Fig. 2.1(a), evolves according to

$$\mathbf{r}(t + \Delta t) = \tanh\left[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)\right], \qquad (2.1)$$

where, for a vector $\mathbf{q} = [q_1, q_2, \dots]^T$, $\tanh(\mathbf{q})$ is the vector $[\tanh(q_1), \tanh(q_2), \dots]^T$.

The R/O coupler is $\mathbf{W}_{out}(\mathbf{r}) = \mathbf{P}_1\mathbf{r} + \mathbf{P}_2\mathbf{r}^2$, where $\mathbf{P_1}$ and $\mathbf{P_2}$ are $D_{out} \times D_r$ matrices, $\mathbf{p} = (\mathbf{P}_1, \mathbf{P}_2)$, and $\mathbf{r}^2$ is the D_r dimensional vector whose j^{th} component is r_j^2. (We found that the simpler choice $\mathbf{W}_{out}(\mathbf{r}) = \mathbf{P}_1\mathbf{r}$ typically did not work for our illustrative example [1].) While, for illustration, we use the specific reservoir dynamics of Eq. (2.1), we emphasize that there is great versatility in the scheme of Fig. 2.1. E.g., for tasks other than prediction, very fast processing has been achieved by using high dimensional photonic systems as the reservoir [24–27] (see also Ref. [28]).

[1]We believe that this may be due to the odd symmetry of the tanh function: With $\mathbf{P}_2 = 0$, the setup in Fig. 2.1(b) is such that, if $\mathbf{r}(t)$ is an attracting reservoir orbit with output $\mathbf{v}(t)$ for $y(x,t)$, then $-\mathbf{r}(t)$ is also an attracting reservoir orbit and corresponds to an output $-\mathbf{v}(t)$. Thus with $\mathbf{P}_2 = 0$ the reservoir dynamics has a symmetry in conflict with the KS equation which is *not* invariant to the change $y \to -y$. Having $\mathbf{P}_2 \neq 0$ breaks this unwanted reservoir symmetry

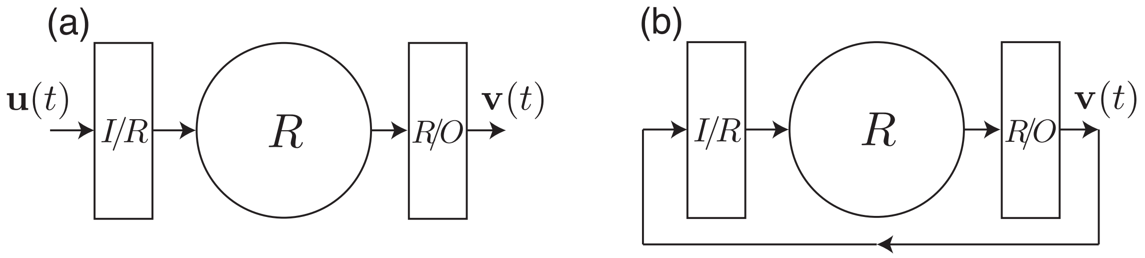


Figure 2.1: I/R : (Input-Reservoir Coupler); R : (Reservoir); R/O : (Reservoir-Output Coupler). (a) Training data gathering phase. (b) Predicting phase. It is assumed that the parameters of the reservoir are chosen such that the "echo state property" is satisfied [1]: all of the conditional Lyapunov exponents of the training reservoir dynamics conditioned on $\mathbf{u}(t)$ are negative so that, for large t, the reservoir state $\mathbf{r}(t)$ does not depend on initial conditions.

In the prediction phase, $t > 0$, $\mathbf{u}(t)$ in Eq. (2.1) is replaced by $\mathbf{v}(t) = \mathbf{W}_{out}(\mathbf{r}(t))$. Regardless of the short-term quality of the predictions $\mathbf{v}(t)$, they will eventually diverge from the true state $\mathbf{u}(t)$ due to the exponential separation of trajectories that is a characteristic of chaotic systems. Consider now the situation where at some future time $\theta > 0$, one wants to predict $\mathbf{u}(t)$ for $t > \theta$ based on measurements of $\mathbf{u}$ up to time θ. The reservoir can then be reinitialized using Eq.(2.1) for a short period of time preceding θ, i.e., $(\theta - \epsilon \leq t \leq \theta)$, to determine $\mathbf{r}(\theta)$, and then used to predict for $t > \theta$. (Once the training is done, it need not be repeated for predictions of subsequent time intervals.)

As an illustrative model for a spatiotemporally chaotic system, we consider the Kuramoto-Sivashinsky (KS) equation modified by the addition (last term in Eq. (2.2)) of a spatial inhomogeneity term,

$$y_t = -yy_x - y_{xx} - y_{xxxx} + \mu \cos\left(\frac{2\pi x}{\lambda}\right). \tag{2.2}$$

The scalar field $y(x,t)$ is periodic in the interval $[0, L)$ and L is an integer multiple of λ. Note that the attractor dimension depends directly on the dimensionless parameter L and scales linearly with L for large L [29]. For later comparison, we note that for $L \geq 100$, the RMS value of y_t is about 0.34, which can be compared to the value of μ to roughly assess the strength of the inhomogeneity on the dynamics. This equation reduces to the standard KS equation when $\mu = 0$. The cosine perturbation breaks the translation symmetry when $\mu \neq 0$. In this chapter, we will consider both $\mu = 0$ and $\mu \neq 0$ in order to probe the effect of spatial homogeneity on our predictions. Equation (2.2) is integrated on a grid of Q equally spaced points with $\Delta t = 0.25$, giving a simulated data set with Q time series, which we denote by the vector $\mathbf{u}(t)$ and use as the reservoir input. Figure 2.2(a) shows our numerical solution of Eq. (2.2) for a KS system with $L = 22, Q = 64$, and $\mu = 0$, while figure 2.2(b) shows a reservoir performed prediction using the scheme described above (Fig. 2.1). Figure 2.2(c) shows the difference between the prediction and the actual solution (we remark that this error metric may overemphasize errors due to spatial shifting of the patterns).

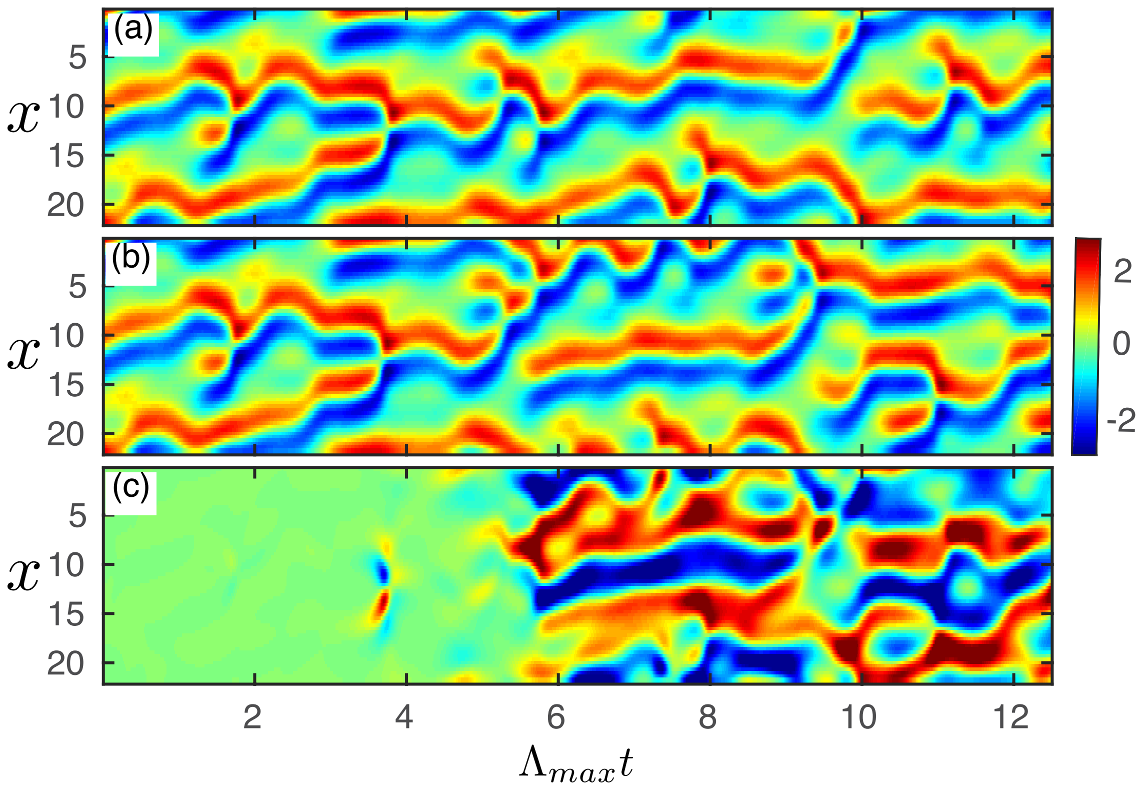


Figure 2.2: Prediction of a KS equation with $L = 22$, $\mu = 0$ using a single reservoir of size $D_r = 5000$. (a) Actual data from the KS model. (b) Reservoir prediction. (c) Error (panel (b) minus panel (a)) in the reservoir prediction. We multiply t by the largest Lyapunov exponent (Λ_{max}) of the model, so that each unit on the horizontal axis represents one Lyapunov time, i.e., the average amount of time for errors to grow by a factor of e.

Although the results of Fig. 2.2 indicate the potential for reservoir-computer-based prediction of spatiotemporal chaos, we note that, as L increases, the size D_r of the reservoir network required to predict the system using a single reservoir (as described by Fig. 2.1) increases. We find that this makes prediction with a single reservoir intractable for much larger values of L. In order to treat large systems, we take advantage of the local nature of interactions in typical spatiotemporally chaotic systems, as was done in Ref. [14] in the context of delay co-ordinates. We propose a parallelized scheme consisting of a large set of reservoirs of moderate size, each of which predicts a local region of the spatio-temporal system. We comment that a somewhat similar structure is employed by convolutional neural networks (CNN's), e.g., see chapter 9 of Ref. [18]. CNN's are widely used in deep learning for image processing tasks, and employ a translationally invariant structure (as we will later discuss for our KS example with $\mu = 0$).

Consider a spatiotemporal system on a one dimensional grid of size Q with periodic boundary conditions, giving us a multivariate data set with Q time series which we denote by the vector $\mathbf{u}(t)$. The Q variables $u_j(t)$ are split into g groups, each group consisting of q spatially contiguous variables such that $gq = Q$. We denote the states of the spatial points in each of the g groups by the vectors $\mathbf{g}_i(t)$: $\mathbf{g}_1(t) = (u_1(t), u_2(t), \cdots u_q(t))^T$, $\mathbf{g}_2(t) = (u_{q+1}(t), u_{q+2}(t), \cdots u_{2q}(t))^T$, and so on. Each group of time series, $\mathbf{g}_i$, is predicted by a reservoir R_i with adjacency matrix $\mathbf{A}_i$, internal state $\mathbf{r}_i(t)$ and input weights $\mathbf{W}_{in,i}$. We denote the input to the i^{th} network by $\mathbf{h}_i(t)$, where $\mathbf{h}_i(t)$ is such that each reservoir accepts inputs from all of the spatial points in the i^{th} group as well as from two contiguous buffer regions of l spatial points on its left and right hand sides, $\mathbf{h}_i(t) = (u_{(i-1)q-l+1}(t), u_{(i-1)q-l+2}(t), \cdots, u_{iq+l}(t))^T$ (the subscript j in u_j is taken modulo Q). Thus, adjacent reservoir networks have overlapping inputs with the size of the overlap set by the locality parameter l (see Fig. 2.3).

The data from $t = -T$ to $t = 0$ is used to train the reservoir network, while the data from $t > 0$ is used to evaluate the quality of the reservoir predictions. Similar to Eq. (2.1), in the training phase, each of the g reservoirs evolves in parallel according to $\mathbf{r}_i(t + \Delta t) = \tanh(\mathbf{A}_i\mathbf{r}_i(t) + \mathbf{W}_{in,i}\mathbf{h}_i(t))$, $1 \le i \le g$, from $t = -T$ to $t = 0$. The g reservoirs are then trained by finding a set of output weights $\mathbf{p}_i = (\mathbf{P}_{1,i}, \mathbf{P}_{2,i})$ for each reservoir such that $\mathbf{P}_{1,i}\mathbf{r}_i(t) + \mathbf{P}_{2,i}\mathbf{r}_i^2(t) \simeq \mathbf{g}_i(t)$. The trained reservoirs with their output weights are now used to predict the time series, $\tilde{\mathbf{g}}_i(t) = \mathbf{P}_{1,i}\mathbf{r}_i(t) + \mathbf{P}_{2,i}\mathbf{r}_i^2(t)$, $\mathbf{r}_i(t+\Delta t) = \tanh(\mathbf{A}_i\mathbf{r}_i(t) + \mathbf{W}_{in}\tilde{\mathbf{h}}_i(t))$, where $\tilde{\mathbf{h}}_i(t)$ is determined from $\tilde{\mathbf{g}}_i(t)$ and the output of the neighboring reservoirs, and we use a superscribed tilde to denote a predicted quantity.

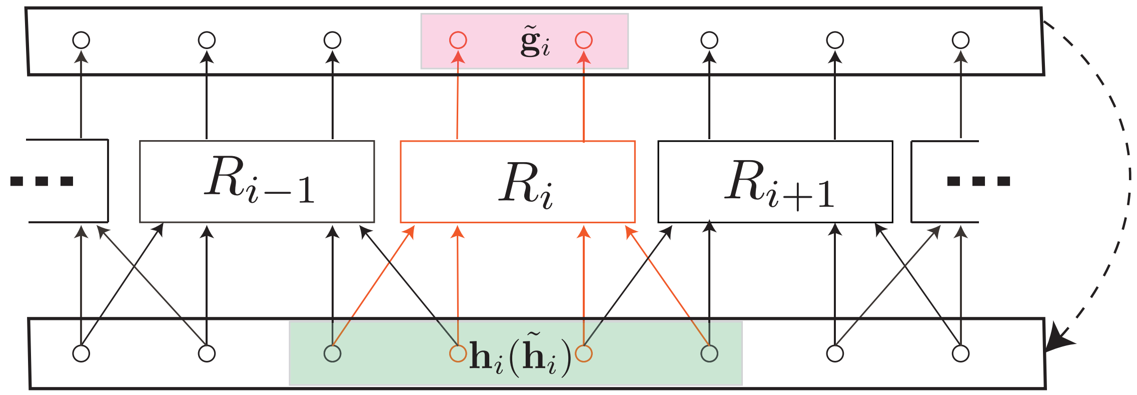


Figure 2.3: Illustration of the parallellized reservoir scheme ($q = 2$, $l = 1$). The pink shaded vector above R_i represents its output $\tilde{\mathbf{g}}_i$. The green shaded vector below R_i represents its input $\mathbf{h}_i$ (during training) and $\tilde{\mathbf{h}}_i$ (during prediction). The dashed arrow shows the feedback connection applied during the autonomous prediction phase ($t \geq 0$).

We now present numerical results; unless otherwise specified, the reservoir parameters used are $D_r = 5000$, $T = 70000$ steps, $\rho = 0.6$, $\sigma = 1.0$, $l = 6$ and $\kappa = 3$. Once the reservoir is trained and the output weights are determined, the resulting autonomous reservoir is used to make a series of predictions, which are then compared with the evaluation data set. We perform predictions on $K = 30$ non-overlapping time intervals, $\theta_k \leq t < \theta_k + \tau$, each of length $\tau = 1000$ in the evaluation data set. Here $\theta_k = (k-1)\tau$ denotes the start of each prediction interval. Before the start of each prediction interval, all reservoir states are reset to $\mathbf{r}_i = \mathbf{0}$ and the reservoirs are then evolved with the true measurements $\mathbf{u}(t)$ for $\epsilon = 10$ time steps, i.e., from $t = \theta_k - \epsilon$ to θ_k, according to $\mathbf{r}_i(t + \Delta t) = \tanh(\mathbf{A}_i\mathbf{r}_i(t) + \mathbf{W}_{in,i}\mathbf{h}_i(t))$, $1 \leq i \leq g$. This gives the reservoir appropriate initial conditions to begin predicting autonomously for the next τ steps. The RMS error between $\mathbf{u}(t)$ and $\tilde{\mathbf{u}}(t) = (\tilde{\mathbf{g}}_1(t), \ldots, \tilde{\mathbf{g}}_g(t))$ is averaged over the K independent predictions to give an estimate of the typical quality of prediction. We perform the same prediction 10 times, for different random reservoir realizations, and calculate the average RMSE over all the trials.

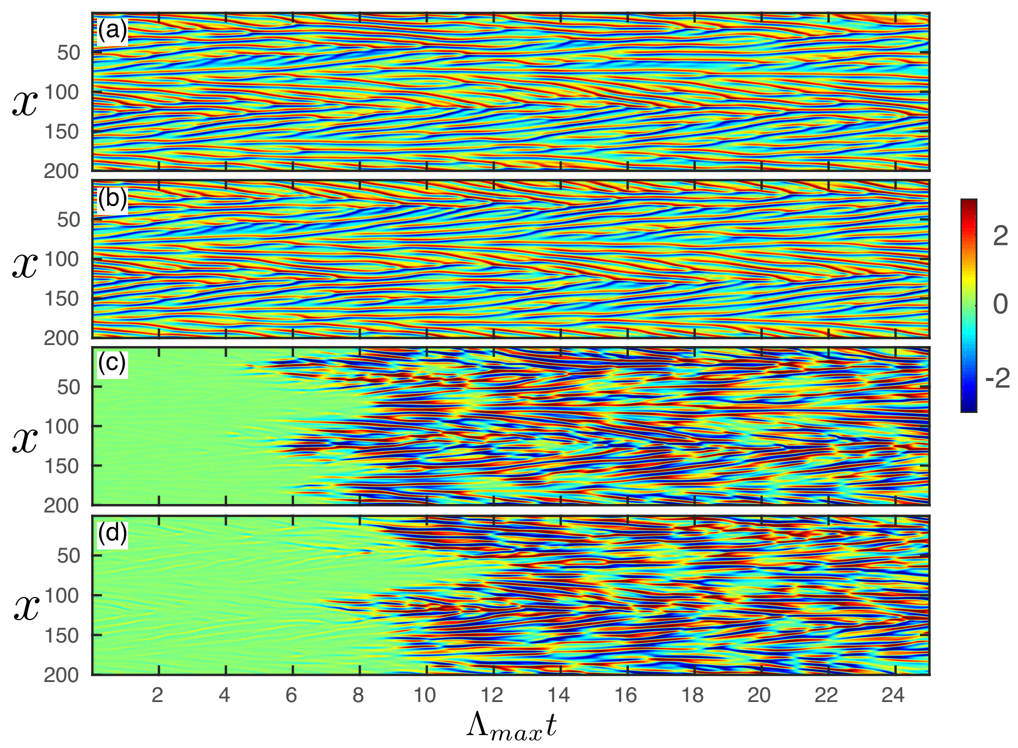


Figure 2.4: Prediction of KS equation ($L = 200$, $Q = 512$, $\mu = 0.01$, $\lambda = 100$) with the parallelized reservoir prediction scheme using $g = 64$ reservoirs. (a) Actual KS equation data. (b) Reservoir prediction ($\tilde{\mathbf{u}}(t)$). (c) Error in the reservoir prediction. (d) Error in a prediction made by integrating the KS equation when it uses the reservoir output at $t = 0$, $\tilde{\mathbf{u}}(0)$, as its initial condition.

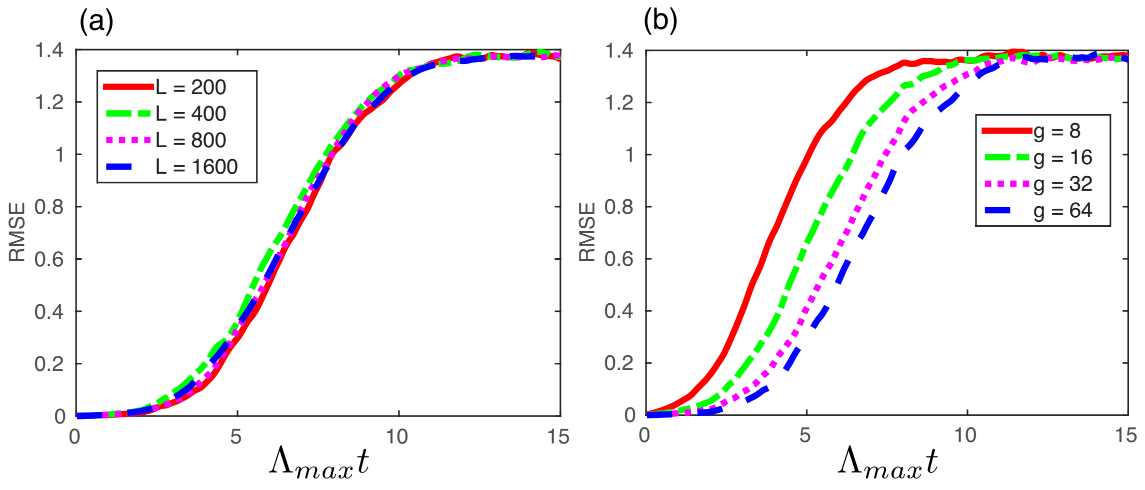


Figure 2.5: (a) RMS error in the predictions of the KS system as function of time for different system sizes $L = 200, 400, 800, 1600$ with L/g held fixed at $200/64$ for all four curves. (b) Improvement of the prediction performance as the number (g) of reservoirs employed is increased; $L = 200$, $Q = 512$, $\mu = 0.01$, $\lambda = 100$.

Figure 2.4 shows the results for a KS equation ($L = 200$, $\mu = 0.01$, $Q = 512$) where panel (a) is the numerical solution of Eq. (2.2), panel (b) is the reservoir prediction using $g = 64$ reservoirs of size $D_r = 5000$ each, and panel (c) is the prediction error (panel (a) minus panel (b)). We see that low prediction error is obtained for about 8 Lyapunov times. As a performance benchmark, panel (d) shows the error of the prediction made by integrating the KS equation (with the same solution method as panel (a)) using the output of the reservoir at $t = 0$ as its initial condition. Thus, panels (c) and (d) have the exact same error at $t = 0$. We see that the prediction time in panel (d) is only slightly longer than that for panel (c), indicating good reproduction of the true dynamics by the reservoir system.

Figure 2.5(a) shows that we can obtain predictions comparable to Fig. 2.4 independent of the system size L. Table 2.1 indicates the largest Lyapunov exponent Λ_{max} and estimated Kaplan-Yorke dimension [30] of the KS system along with the number of reservoirs (g) and the total number of nodes N_T in the g reservoirs used for Fig. 2.5(a).

L	Λ_{max}	D_{KY}	g	N_T $(\times 10^5)$
100	0.09	23	32	1.6
200	0.09	43	64	3.2
400	0.09	85	128	6.4
800	0.1	167	256	12.8
1600	0.1	338	512	25.6

Table 2.1: Largest Lyapunov Exponent (Λ_{max}) and Kaplan-Yorke Dimension (D_{KY}) of the attractor ($\lambda = 100, \mu = 0.01$) along with the number of parallel reservoirs (g) and the total number (N_T) of all nodes in the g reservoirs of the parallelized reservoir scheme used.

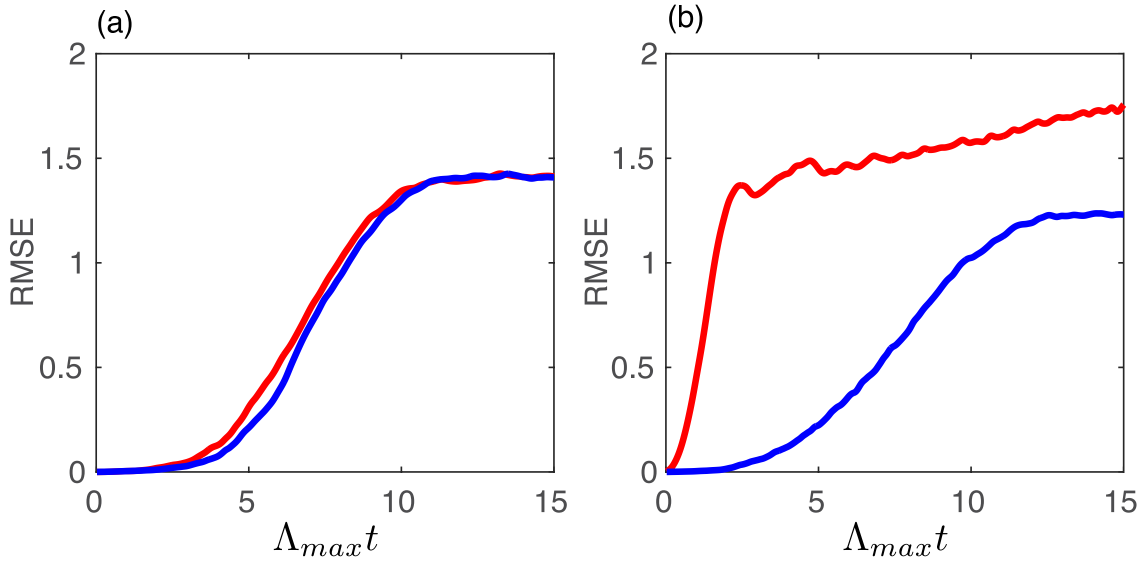


Figure 2.6: Reservoir prediction performance for the KS equation with $L = 200, \lambda = 100$ (a): $\mu = 0$ and (b) $\mu = 0.01$. The red curve shows the RMSE curve when all $g = 64$ reservoirs are identical and have the same output weights. The blue curve shows the RMSE when the g parallel reservoirs are independently trained.

When the strength of the cosine perturbation term is set to $\mu = 0$, the KS equation (Eq. (2.2)) has translation symmetry which can be exploited to drastically reduce the computational cost of training the output weights. We find that it is then sufficient to train a single reservoir (say R_1) by evolving it according to $\mathbf{r}_1(t+\Delta t) = \tanh(\mathbf{A}_1\mathbf{r}_1(t)+\mathbf{W}_{in,1}\mathbf{h}_1(t))$ and then calculating $(\mathbf{P}_{1,1}, \mathbf{P}_{2,1})$. We then use g identical reservoir systems with $\mathbf{W}_{in,i} = \mathbf{W}_{in,1}$, $\mathbf{A}_i = \mathbf{A}_1$, and $(\mathbf{P}_{1,i}, \mathbf{P}_{2,i}) = (\mathbf{P}_{1,1}, \mathbf{P}_{2,1})$ for $1 \leq i \leq g$ in the prediction phase equations. As shown by the agreement between the red (identical weights) and blue (individually trained weights) curves in Fig 2.6(a), this works well. However, when $\mu = 0.01$, the method of identical weights fails as expected (Fig. 2.6(b)). Note that the Lyapunov spectrum for $\mu = 0.01$ is very close to the spectrum for $\mu = 0$ (see supplement).

Further details on the specific reservoir computer parameters, implementation and methods are given in the supplemental material of Ref. [31] which includes Refs. [32,33]. The additional material illustrates that the performance shown above is very robust, in that it changes little over wide ranges in the various parameters.

In conclusion, our results suggests that machine learning, and in particular reservoir computing, offers an effective potential means for model-free prediction of large spatiotemporally chaotic systems.

Chapter 3: Using Machine Learning for Data-Driven Analysis of Dynamical Systems

This chapter is based on the paper, 'Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data', Chaos 27, 121102 (2017); https://doi.org/10.1063/1.5010300 by Jaideep Pathak, Zhixin Lu, Brian R. Hunt, Michelle Girvan, and Edward Ott. ©by the American Institute of Physics

We consider the frequently occurring situation in which limited duration time series data from some dynamical process is available, but a first-principles-based model of how that data is produced is either unavailable or too inaccurate to be useful. Thus, if one is interested in diagnosing ergodic properties of the underlying processes producing the data, one is restricted to do so based only on the data itself. We call such a method "model-free." Model-free analysis of dynamical time series is a long-standing subject of study in nonlinear dynamics [34–36]. Perhaps the most wide-spread approach uses delay-coordinate embedding [34–42]. In this article, we discuss a very promising, entirely different approach to model-free analysis of dynamical time series. Our approach is based upon recent significant advances in the area known as *machine learning*. In particular, we will apply a type of machine learning known as reservoir computing [43], and, for definiteness, we focus on the problem of determining the Lyapunov exponents of the data-generating system. For this application, the key ability we require from machine learning is to replicate the ergodic properties of the system generating the input, and we call this replicating the "climate."

The rest of this article is organized as follows. Section 3.1 reviews reservoir computing and its use for short-term prediction of chaotic time series. Section 3.2 illustrates our method using the well-known Lorenz 1963 model [2], and discusses the ability of reservoir computers to replicate the (long-term) climate. Section 3.3 uses our approach to evaluate the Lyapunov exponents of the Kuramoto-Sivashinsky (KS) equation [44–46] with periodic boundary conditions. This system provides an example of extensive spatiotemporal chaos [47–50], for which the attractor dimension and number of positive Lyapunov exponents increases linearly with the periodicity length L. In particular, Sec. 3.3 considers cases with many positive Lyapunov exponents. The chapter concludes with further discussion in Sec. 3.4.

The main conclusion of this chapter is that our machine learning approach offers a very attractive model-free method for obtaining Lyapunov exponents from data. Particularly notable are our results from Sec. 3.3 where we obtain excellent agreement for all of the positive Lyapunov exponents and many of the negative exponents for a moderately high-dimensional system. In comparison with delay coordinate embedding, we remark that our method appears to be simpler to implement, and does not appear to suffer from the problem of yielding spurious positive Lyapunov exponents [[51]],[[52]] (these papers and references therein discuss a mechanism responsible for spurious positive Lyapunov exponents in delay coordinate embedding; this mechanism is inherently absent in our method). More broadly, our chapter suggests that machine learning is useful for analysis of data from chaotic systems (e.g., previous work has treated model-free machine learning for prediction of future evolution of the states of a dynamical system [53] and for inference of unmeasured dynamical variables [22]).

3.1 Reservoir Computers, Short Term Prediction and Attractor Climate

Reservoir computers [43] originate from an idea independently put forth about 16 years ago in two papers [54,55]. The general approach is illustrated in Fig. 3.1(a), which shows an input vector $\mathbf{u}(t)$ fed into a "reservoir" (labeled R in Fig. 3.1(a)) through an input-to-reservoir coupler (labeled I/R), with an output vector $\mathbf{v}$ coupled from the reservoir through an output coupler (labeled R/O). We regard the couplers as acting instantaneously and without memory (i.e., their output depends solely on their current input). Importantly, the reservoir has memory (i.e., it has internal dynamics so its state depends on its history). We assume that it receives input at discrete times t, and that its input $\mathbf{W}_{in}\mathbf{u}(t)$ is combined with the reservoir state $\mathbf{r}(t)$ to produce its output $\mathbf{r}(t + \Delta t)$. In general, the reservoir is an appropriate complex dynamical system; here we follow Refs. [[54], [55]] and consider the reservoir to be a large random network with D_r nodes and an $D_r \times D_r$ adjacency matrix $\mathbf{A}$. Specifically, we will henceforth consider the particular implementation (similar to

Ref. [[53]]) of Fig. 3.1(a) given by

$$\mathbf{r}(t+\Delta t) = \tanh[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)], \tag{3.1}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{W}_{out}(\mathbf{r}(t+\Delta t), \mathbf{P}), \tag{3.2}$$

where $\mathbf{r}(t)$ represents the scalar states $r_i(t)$ of the D_r network reservoir nodes, $\mathbf{r} = [r_1, r_2, ..., r_{D_r}]^T$; in Eq. (3.1), $\mathbf{W}_{in}$ is a $D_r \times D$ matrix, where D is the dimension of $\mathbf{u}$; also, in Eq. (3.1), for a vector $\mathbf{q} = (q_1, q_2, \dots)^T$ the quantity $\tanh(\mathbf{q})$ is the vector $(\tanh(q_1), \tanh(q_2), \dots)^T$. In Eq. (3.2), $\mathbf{W}_{out}$ maps the D_r dimensional vector $\mathbf{r}$ to the output $\mathbf{v}$, which, for the situations considered in this article, has the same dimension D as $\mathbf{u}$. In addition, we assume that $\mathbf{W}_{out}$ depends on a large number of adjustable parameters given by the elements of the matrix $\mathbf{P}$, and that $\mathbf{W}_{out}(\mathbf{r}, \mathbf{P})$ depends linearly on $\mathbf{P}$ (e.g., in past work the choice $\mathbf{W}_{out}(\mathbf{r}, \mathbf{P}) = \mathbf{P}\mathbf{r}$ has often been used).

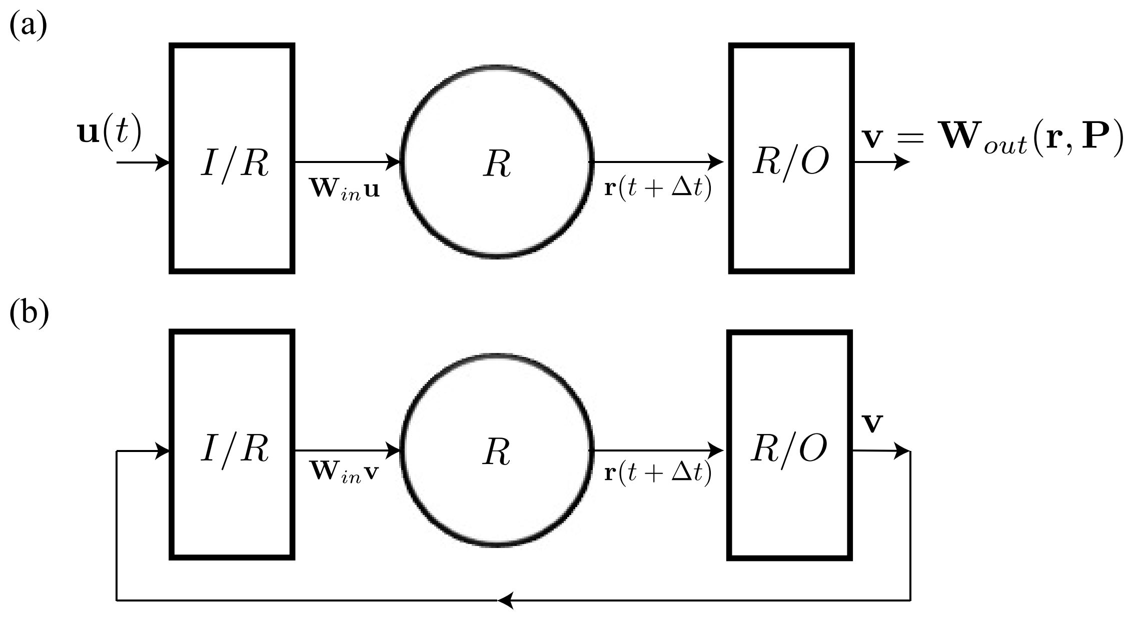

(a)
$\mathbf{u}(t)$
I/R
$\mathbf{W}_{in}\mathbf{u}$
R
$\mathbf{r}(t+\Delta t)$
R/O
$\mathbf{v} = \mathbf{W}_{out}(\mathbf{r}, \mathbf{P})$
(b)
I/R
$\mathbf{W}_{in}\mathbf{v}$
R
$\mathbf{r}(t+\Delta t)$
R/O
$\mathbf{v}$

Figure 3.1: (a) Configuration of the reservoir in the training phase corresponding to Eqs. 3.1 and 3.2. (b) Reservoir configuration in the prediction phase corresponding to Eq. 3.4. I/R and R/O denote the input-to-reservoir and reservoir-to-output couplers respectively. R denotes the reservoir.

In general, the goal of the system in Fig. 3.1(a) is for the outputs $\mathbf{v}(t)$ to approximate the desired outputs, $\mathbf{v}_d(t)$, appropriate to the inputs $\mathbf{u}(t)$ (e.g., in a pattern recognition task $\mathbf{u}(t)$ might represent a sequence of patterns, and $\mathbf{v}_d(t)$ would represent classifications the patterns). To this end, during a training period, $-T \leq t \leq 0$, an input $\mathbf{u}(t)$ is fed into the reservoir and the resulting reservoir state evolution $\mathbf{r}(t)$, along with $\mathbf{u}(t)$, are recorded and stored as "training data." Then the parameters $\mathbf{P}$ are chosen ("trained") so as to approximately minimize the mean squared difference between $\mathbf{v}(t)$ and its desired value $\mathbf{v}_d(t)$. As is common in reservoir computing, we use the Tikhonov regularized regression procedure [32] to find an output matrix $\mathbf{P}$, that minimizes the following function,

$$\sum_{-T\leq t\leq 0} ||\mathbf{W}_{out}(\mathbf{r}(t),\mathbf{P}) - \mathbf{v}_d(t)||^2 + \beta||\mathbf{P}||^2, \tag{3.3}$$

where $||\mathbf{P}||^2$ denotes the sum of the squares of elements of $\mathbf{P}$. The regularization constant $\beta > 0$ discourages overfitting by penalizing large values of the fitting parameters (In Sec. 3.3 we used a value $\beta > 0$, but for Sec. 3.2 we found that using $\beta = 0$ was sufficient). For a given task, one hopes that for large enough D_r and T, the system in Fig. 3.1(a) will yield subsequent ($t > 0$) outputs $\mathbf{v}(t)$ that closely approximate the desired $\mathbf{v}_d(t)$. Because $\mathbf{W}_{out}(\mathbf{r},\mathbf{P})$ is taken to be linear in $\mathbf{P}$, the problem of determining the parameters $\mathbf{P}$ that minimize Eq. (3.3) is one of linear regression for which there are well-established techniques [23]. This approach has been shown to work extremely well for a wide variety of tasks [43].

We now consider the task of prediction for the case where $\mathbf{u}(t)$ depends on the state of some deterministic dynamical system. This problem was originally considered in the reservoir computer framework by Jaeger and Haas [53]. The idea is to take the desired output to be the same as the input, $\mathbf{v}_d(t+\Delta t) = \mathbf{u}(t+\Delta t)$. When one wishes to commence prediction at $t = 0$, the configuration is switched from that in Fig. 3.1(a) to that in Fig. 3.1(b), and the reservoir system is run autonomously according to the following equation.

$$\mathbf{r}(t+\Delta t) = \tanh\left[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{W}_{out}(\mathbf{r}(t),\mathbf{P})\right]. \tag{3.4}$$

The output of the autonomous reservoir, $\mathbf{v}(t) = \mathbf{W}_{out}(\mathbf{r}(t), \mathbf{P})$, gives the predicted value $\mathbf{u}(t)$ for $t > 0$. Jaeger and Haas [53], using the example of the Lorenz system [2], indeed verified that this prediction scheme works and gives good short term predictions. As expected, the chaotic amplification of small errors leads to eventual breakdown of the prediction, limiting the prediction time. However, as shown in the next two sections, following this breakdown of short-term prediction, the evolution of $\mathbf{v}(t)$ often provides an accurate approximation for the climate corresponding to $\mathbf{u}(t)$, and can be used in particular to compute Lyapunov exponents of the process that generated $\mathbf{u}(t)$.

3.2 Climate Replication in the Lorenz System

In this section we illustrate the capability of our technique to replicate the "climate" of the Lorenz 1963 system [2],

$$\dot{x} = 10(y - x), \dot{y} = x(28 - z) - y, \dot{z} = xy - 8z/3. \tag{3.5}$$

We construct and train reservoir computers with input $\mathbf{u} = (x, y, z)^T \in \mathbb{R}^3$ and output $\mathbf{v} \in \mathbb{R}^3$, following Sec. 3.1. The reservoir network is built from a sparse random Erdős-Rényi network whose average degree is $d = 6$. Each non-zero element in the adjacency matrix is drawn independently and uniformly from $[-a, a]$, and $a > 0$ is adjusted so that the spectral radius of $\mathbf{A}$ (the largest magnitude of its eigenvalues) has a desired value ρ. During the training phase, $-T \leq t \leq 0$ (where $T = 100$), the reservoir computer evolves following Eq. (3.1) with $\Delta t = 0.02$. In

Parameter	Value	Parameter	Value
D_r	300	d	6
T	100	Δt	0.02
$T/\Delta t$	5000	β	0
ρ	1.2	σ	0.1

Table 3.1: Standard reservoir parameters used for a successful climate replication of the Lorenz system (referred to in the text as the $R1$ reservoir). The $R2$ reservoir uses the same parameters with a different spectral radius, $\rho = 1.45$.

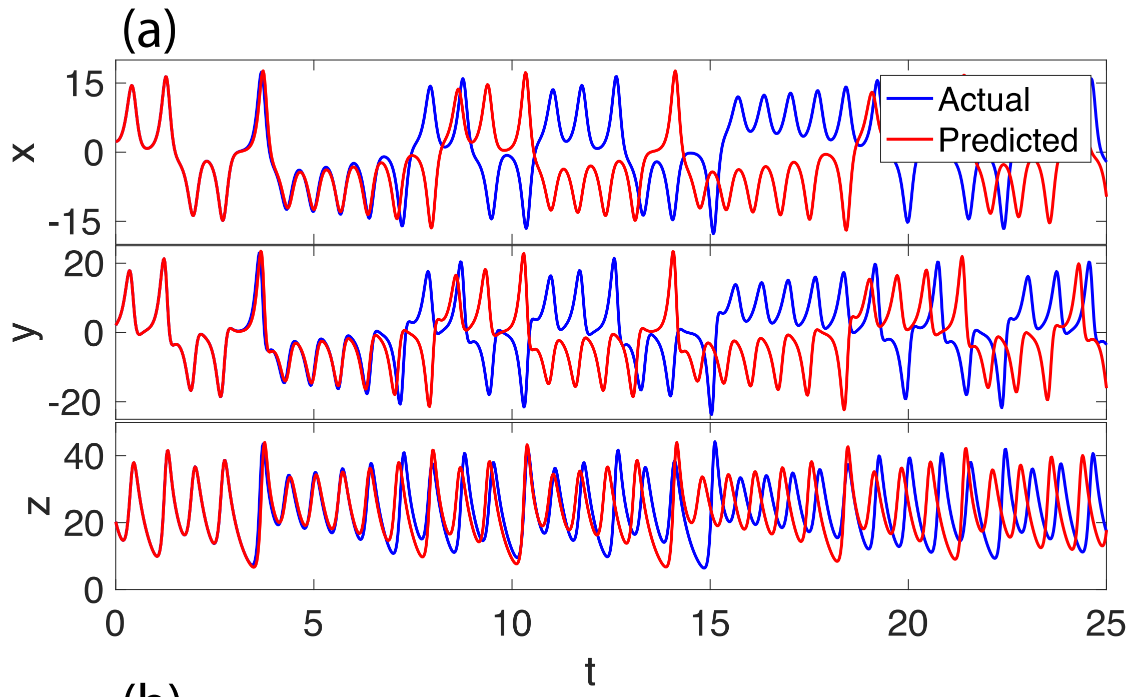
this Lorenz example, the reservoir output $\mathbf{v}(t) = \mathbf{W}_{out}(\mathbf{r}(t), \mathbf{P})$ is defined as

$$\mathbf{v}(t) = \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1\mathbf{r}(t) \\ \mathbf{p}_2\mathbf{r}(t) \\ \mathbf{p}_3\tilde{\mathbf{r}}(t) \end{bmatrix} \tag{3.6}$$

where $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ are the rows of the $3 \times D_r$ matrix $\mathbf{P}$. The quantity $\tilde{\mathbf{r}}$ in the third line of Eq. (5.2) is defined in a way such that the first half of its elements are the same as that of $\mathbf{r}$, i.e., $\tilde{r}_i = r_i$ for half $(D_r/2)$ of the reservoir nodes, while $\tilde{r}_i = r_i^2$ for the remaining half of the reservoir node (Our use here of $\tilde{\mathbf{r}}(t)$, rather than $\mathbf{r}(t)$, to predict $z(t)$ is related to the $x \to -x$, $y \to -y$ symmetry of the Lorenz equations as discussed in Ref. [22]).

After we compute $\mathbf{r}(t)$ for the training period, $-T \leq t \leq 0$, we calculate the output weight parameters $\mathbf{P}$ that minimize the function in Eq. (3.3) with the desired output being the state variables from the Lorenz system, $\mathbf{v}_d(t) = [x(t), y(t), z(t)]^T$ (in an actual physical experiment, we assume $\mathbf{u}(t) = \mathbf{v}_d(t)$ to have been measured for $-T \leq t \leq 0$). After we find the output weights, we evolve the reservoir with the reconfigured reservoir system (Fig. 3.1(b)).

Following the above described procedure, We now report and compare results for two simulations using reservoir configurations with $\rho = 1.2$ (denoted R1) and $\rho = 1.45$ (denoted R2). The prediction for $0 < t \leq 25$ for both trained reservoirs are shown in Fig. 3.2(a) (R1 with $\rho = 1.2$) and Fig. 3.2(b) (R2 with $\rho = 1.45$). Both reservoirs R1 and R2 generate correct short-term predictions and then deviate from the actual Lorenz trajectories, which is expected since any small error grows exponentially due to the chaotic dynamics of the Lorenz system. However, after the failure of the short-term prediction, the two reservoirs show qualitatively different dynamical patterns. In Fig. 3.2(a), it seems that, after $t \approx 7$, although the R1 prediction deviates from the actual trajectory, the long-term dynamics appears to resemble that of the original Lorenz system. In contrast, as shown by Fig. 3.2(b), this is clearly not the case for R2.



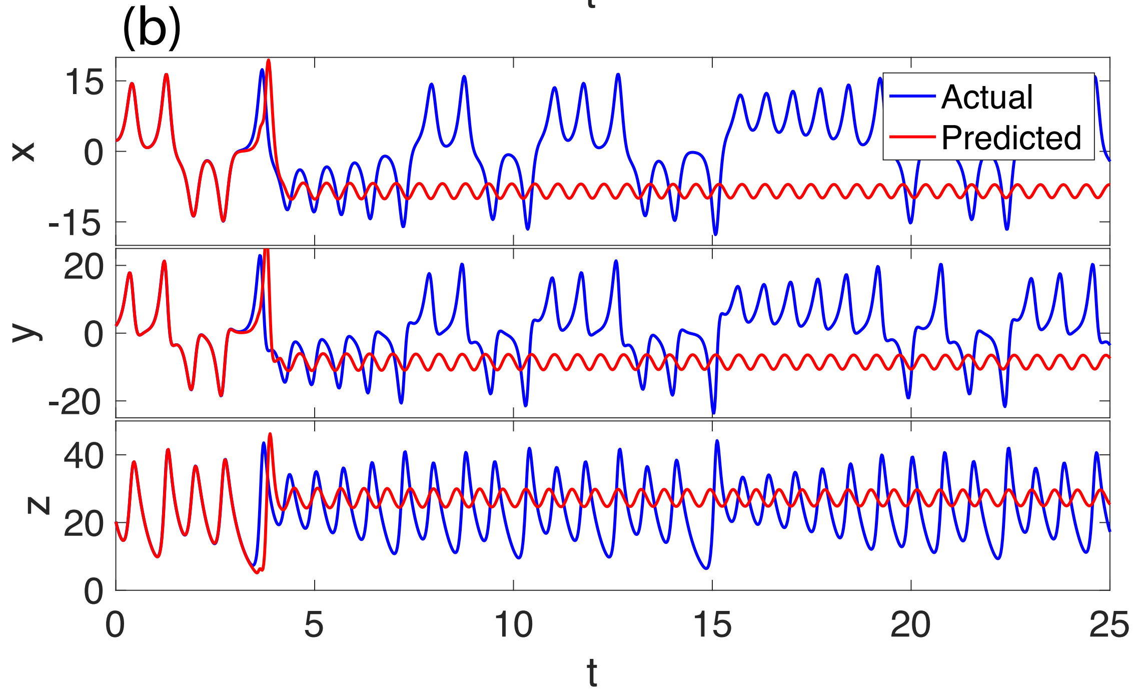


Figure 3.2: (a) The state prediction (red) of the R1 reservoir and the actual trajectories (blue) of the Lorenz system for $0 < t \leq 25$. The spectral radius of the reservoir is 1.2. (b) The state prediction (red) of the R2 reservoir and the actual trajectories (blue) of the Lorenz system for $0 < t \leq 25$. The spectral radius of the reservoir is 1.45.

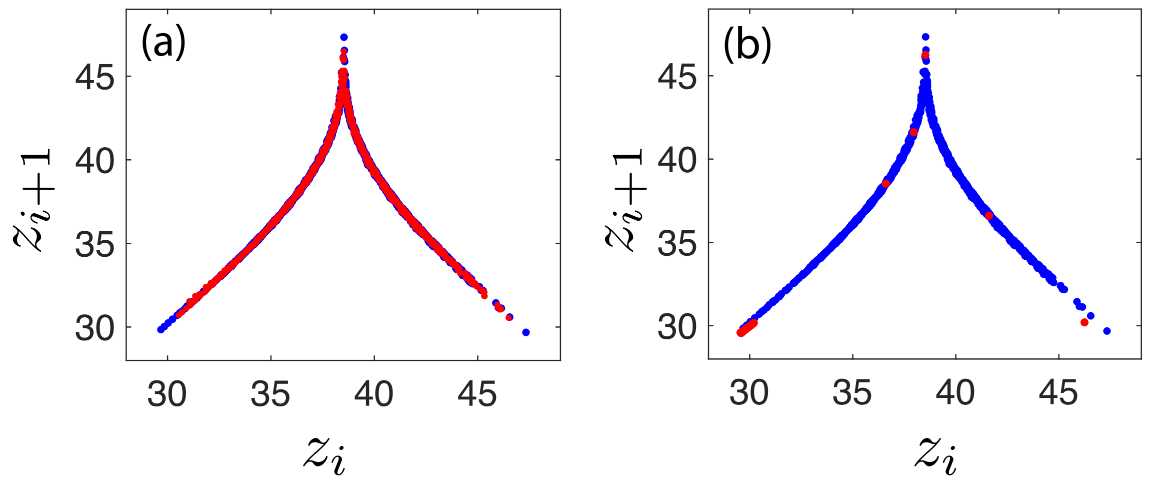


Figure 3.3: The return map of the actual and the predicted z-coordinate of the Lorenz system. This plot is made with time series of length 1000, where the blue dots are from the actual Lorenz system, and the red dots overlaying the blue dots are from the prediction. The left panel shows the return map of the long term prediction of the R1 reservoir with $\rho = 1.2$, while the right panel is from the R2 reservoir with $\rho = 1.45$.

In Fig. 3.3 we present a more accurate test than visual inspection of Figs. 3.2(a) and 3.2(b) for correctness of the climate. To do this, we follow Lorenz's procedure of plotting the return map of successive maxima of $z(t)$. We first obtain $z(t)$ for a long period of time, $0 < t < 1000$, for both the actual and the predicted time series. We then locate all local maxima of the actual and predicted $z(t)$ in time order and denote them $[z_1, z_2, ..., z_m]$. Then, we plot consecutive pairs of those maxima $[z_i, z_{i+1}]$ for $i = 1, ..., m - 1$ as dots in Figs. 3.3. The blue dots in both panels of Figs. 3.3 are from the actual Lorenz system, while the red dots *printed over* the blue dots are from the reservoir output prediction (v_3) of $z(t)$. As confirmed by Fig. 3.3(a) the red dots produced by the R1 reservoir continue to fall on top of the blue dots (from the actual Lorenz system) throughout the entire run time ($0 < t < 1000$). In contrast, Fig. 3.3(b) shows that the blue dots remain largely uncovered, because, as indicated in the third panel of Fig. 3.2(b), the maximum value of $z(t)$ for $t > 5$ is at a fixed point $z_{max} \approx 30$. Thus the R1 reservoir very accurately succeeds in reproducing the long-time climate of the attractor, while the R2 reservoir does not, and this is so even though both setups are apparently capable of producing useful short term predictions. (We have also obtained similar results for many other simulations.) Thus some parameter adjustment may be necessary to avoid unsuccessful reproduction of the climate. Fortunately, we usually find that when the climate is not reproduced it is fairly evident (as in Fig. 3.2(b), as well as Fig. 3.5 of the next section). More quantitatively, a promising general means of assessing whether the reservoir system has succeeded in mimicking the climate is to first use the training data to obtain finite-time estimates of the system's ergodic properties (e.g., frequency-power spectra, time correlations, moments etc.). Once this is done, one can test whether those estimates are consistent with determinations of the same quantities obtained from the long-term reservoir dynamics. Section 3.3 provides such an assessment for the Kuramoto-Sivashinsky system.

	Actual Lorenz System	R1 System	R2 System
Λ_1	0.91	0.90	0.01
Λ_2	0.00	0.00	-0.1
Λ_3	-14.6	-10.5	-9.9

Table 3.2: Three largest Lyapunov exponents $\Lambda_1 \geq \Lambda_2 \geq \Lambda_3$ for the Lorenz system (Eq. (3.5)), and for the reservoir set up in the configuration of Fig. 3.1(b) for R1 and R2. Since the reservoir system that we employ is a discrete time system, while the Lorenz system is a continuous system, for the purpose of comparison, Λ_1, Λ_2, and Λ_3 are taken to be per unit time; that is, their reservoir values (columns 2 and 3) are equal to the reservoir Lyapunov exponents calculated on a per iterate basis divided by Δt.

The reservoir in the autonomous configuration of Fig. 3.1(b) represents a known discrete-time, D_r-dimensional dynamical system (since we know $\mathbf{W}_{in}$, $\mathbf{A}$, and the output parameters $\mathbf{P}$ determined by the training). We compute the equations for the evolution of the tangent map corresponding to Eq. (3.4) and evolve a set of m mutually orthogonal tangent vectors $\mathbf{R}(t) = \{\delta \mathbf{r}_j\}_{j=1}^{m}$ along with Eq. (3.4). We then compute the largest m Lyapunov exponents of the reservoir dynamical system in the the configuration shown in Fig. 3.1(b) using a standard algorithm based on QR decomposition (e.g., see Ref. [[36]]) of the matrix $\mathbf{R}(t)$. The two right-most columns of Table 3.2 show the three largest Lyapunov exponents, $\Lambda_1 \geq \Lambda_2 \geq \Lambda_3$, of the reservoir system in the autonomous configuration, Fig. 3.1(b), for the R1 reservoir (for which climate reproduction succeeds), and for the R2 reservoir (for which climate reproduction fails).

Comparing the Lyapunov exponents of the Lorenz system (first column of Table 3.2) with those of the R1 reservoir, we see that the largest Lyapunov exponent of the R1 reservoir is a good approximation to the largest Lyapunov exponent of the Lorenz system. Also, consistent with the small value of Δt, the reservoir dynamics approximates that of a flow for which Λ_2 should be (and is) approximately zero. On the other hand, we see that the third Lyapunov exponent of the R1 system is less negative than the negative Lyapunov exponent of the true Lorenz system. In contrast with the good agreement of the Λ_1 values for the Lorenz system and the R1 reservoir, the positive Lyapunov exponent of the Lorenz system fails to be reproduced by the R2 system whose largest Lyapunov exponent is approximately zero; this is consistent with the observation from Fig. 3.2(b) that the long term reservoir attractor for R2 appears to be a periodic orbit.

The significant conclusion from the above is that the R1 system, as a result of successfully reproducing the climate, can be utilized to obtain an approximation to the positive and zero Lyapunov exponents of the process generating its input. We note, however, that the R1 system does not accurately reproduce the true negative Lyapunov exponent of the Lorenz attractor.

The inaccurate R1 reservoir estimation of Λ_3, noted above, can be understood by noting that, although the return map in Fig. 3.3 appears to be a curve, this apparent "curve" must, as noted by Lorenz [2], actually have some small width. The R1 reservoir succeeds in *approximating* the attractor of the Lorenz system as reflected by its apparent good reproduction of the return map shown in Fig. 3.3(a). In order to do this, however, the reservoir need not reproduce the very thin transverse structure within the apparent curve. Since, this very thin structure, as we next discuss, is the primary orbital evidence of the value of Λ_3, one might not expect the reservoir to accurately reproduce this very negative Lyapunov exponent. Specifically, using the Kaplan-Yorke formula for the information dimension [30] of the fractal Lorenz attractor, we obtain a dimension of $[2 + (\Lambda_1/|\Lambda_3|)] = 2.06$, corresponding to 1.06 for the dimension of the structure in the return map (Fig. 3.3(a)). This dimension is very close to one, in agreement with the approximate curve-like character of the return map. However, close examination of the return map "curve" of the Lorenz attractor has previously shown that, within its thickness, there is a fractal set of small transverse dimension (presumably $\Lambda_1/|\Lambda_3| = 0.06$). On the other hand, the Kaplan-Yorke dimension for the return map for the climate of the R1 reservoir attractor is about 1.09. Since the primary orbital difference reflected by differing values of Λ_3 is the difference in very thin structure features of the return map that have only a small effect on the climate dynamics, it is not surprising that the R1 reservoir, while giving a good approximation to the true climate of the Lorenz system, gives only a rough approximation of Λ_3.

3.3 Determining a Large Number of Lyapunov Exponents of a High Dimensional Spatiotemporal Chaotic System from Data

We now consider a modified version of the Kuramoto-Sivashinsky (KS) system defined by the partial differential equation for the function $y(x, t)$

$$y_t = -yy_x - \left[1 + \mu \cos\left(\frac{2\pi x}{\lambda}\right)\right] y_{xx} - y_{xxxx}, \tag{3.7}$$

in the region $0 \leq x < L$ with periodic boundary conditions, $y(x, t) = y(x + L, t)$, and λ chosen so that L is an integer multiple of λ. This equation reduces to the standard KS equation when $\mu = 0$. The cosine term makes the equation spatially inhomogeneous. We will subsequently consider the cases $\mu = 0$ and $\mu \neq 0$ in order to discuss the effect of the symmetries of the KS equation on the learning dynamics of the reservoir computer.

By numerically integrating Eq. (3.7) on an evenly spaced one-dimensional grid of size Q, we obtain a discretized multivariate data set of Q time series,

$$\mathbf{u}(t) = \left[y(\Delta x, t), y(2\Delta x, t), \ldots, y(Q\Delta x, t)\right]^T, \Delta x = L/Q.$$

As in the case of the Lorenz equations discussed in Sec. 3.2, we consider the situation where we have access to the time series data but do not have information about the dynamical equation that generated the time series. In the absence of a model, we will use the data to train a reservoir computer to emulate the behavior of the true dynamical system, in this case Eq. (3.7).

Parameter	Value	Parameter	Value
D_r	9000	d	3
T	20000	Δt	0.25
$T/\Delta t$	80000	β	0.0001
ρ	0.4	σ	0.5

Table 3.3: Reservoir parameters used for the successful replication of the climate of the Kuramoto-Sivashinsky system shown in Fig. 3.4.

The reservoir network is as described in Sec. 3.1 with the parameters listed in Table 3.3. In the training phase, Fig. 3.1(a), we evolve the reservoir according to Eq. (3.1) from $t = -T$ to $t = 0$. Next, we use Tikhonov regularized regression (see Eq. (3.3)) to compute the output parameters, $\mathbf{P}$ such that $\mathbf{W}_{out}(\mathbf{r}, \mathbf{P}) = \mathbf{P}\tilde{\mathbf{r}}(t) \simeq \mathbf{u}(t)$ for $-T \leq t < 0$. Here $\tilde{\mathbf{r}}$ is a D_r-dimensional vector such that the i^{th} component of $\tilde{\mathbf{r}}$ is $\tilde{r}_i = r_i$ for half the reservoir nodes and $\tilde{r}_i = r_i^2$ for the remaining half. With the output parameters determined, we let the reservoir evolve autonomously for $t > 0$ as shown in Fig. 3.1(b) according to Eq. (3.4).

The predictions made by the reservoir system for $t > 0$ are given by, $\mathbf{W}_{out}(\mathbf{r}(t), \mathbf{P})$. Figure 3.4 shows the time evolution of one such reservoir prediction for $t > 0$ (middle panel), along with the true state (top panel) of the KS equation and the deviation (bottom panel) of the reservoir prediction from the true state (i.e., the difference between the top panel and the middle panel) Note that in Fig. 3.4 time (the horizontal axis) is in units of the Lyapunov time (Λ_1^{-1}, where Λ_1 is the largest Lyapunov exponent of the KS attractor). We see that the reservoir gives good short term prediction for about 5 multiples of the Lyapunov time. A visual inspection of Fig. 3.4 suggests that the reservoir prediction may have also learned the correct 'climate' of the KS system even after the state of the reservoir dynamical system has diverged from the true state of the KS system.

Figure 3.5 shows an example of an alternate scenario for another set of the reservoir parameters ($\rho = 3.1$, $D_r = 5000$ with the rest of the parameters as shown in Table 3.3). In this case, the reservoir still predicts accurately for a short period of time. However, the long term climate of the signal generated by the reservoir is no longer similar to that of the true KS climate.

A more quantitative assessment of the climate reproduction can be obtained by calculating the power spectrum of the reservoir prediction and comparing it with the power spectrum of the training data. Figure 3.6 shows the power spectrum of the training data, along with the power spectrum of the dynamics of the autonomous reservoir system in Figs. 3.4 and 3.5. We see that the reservoir system corresponding to Fig. 3.4 succeeds in reproducing the training data power spectrum, thus indicating that the long term system orbit reproduces the climate of the training data. On the other hand, the power spectrum of the reservoir system corresponding to Fig. 3.5 confirms our visual assessment that this reservoir system fails to reproduce the climate of the training data.

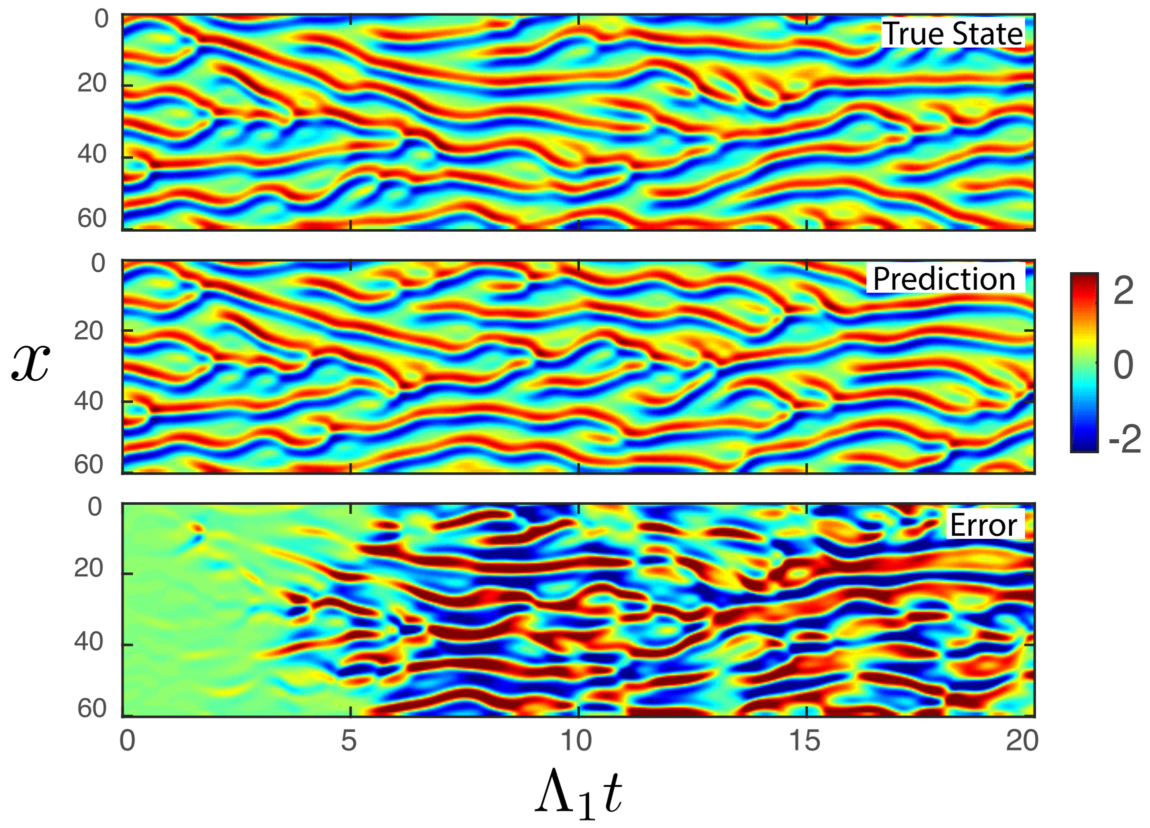


Figure 3.4: Top panel: True state, $y(x, t)$, of the standard KS system after $t = 0$. Middle panel: Reservoir prediction. Bottom panel: Difference between the true state and the reservoir prediction. The parameters of the KS equation are $L = 60$, $\mu = 0$. Λ_1 denotes the largest Lyapunov exponent.

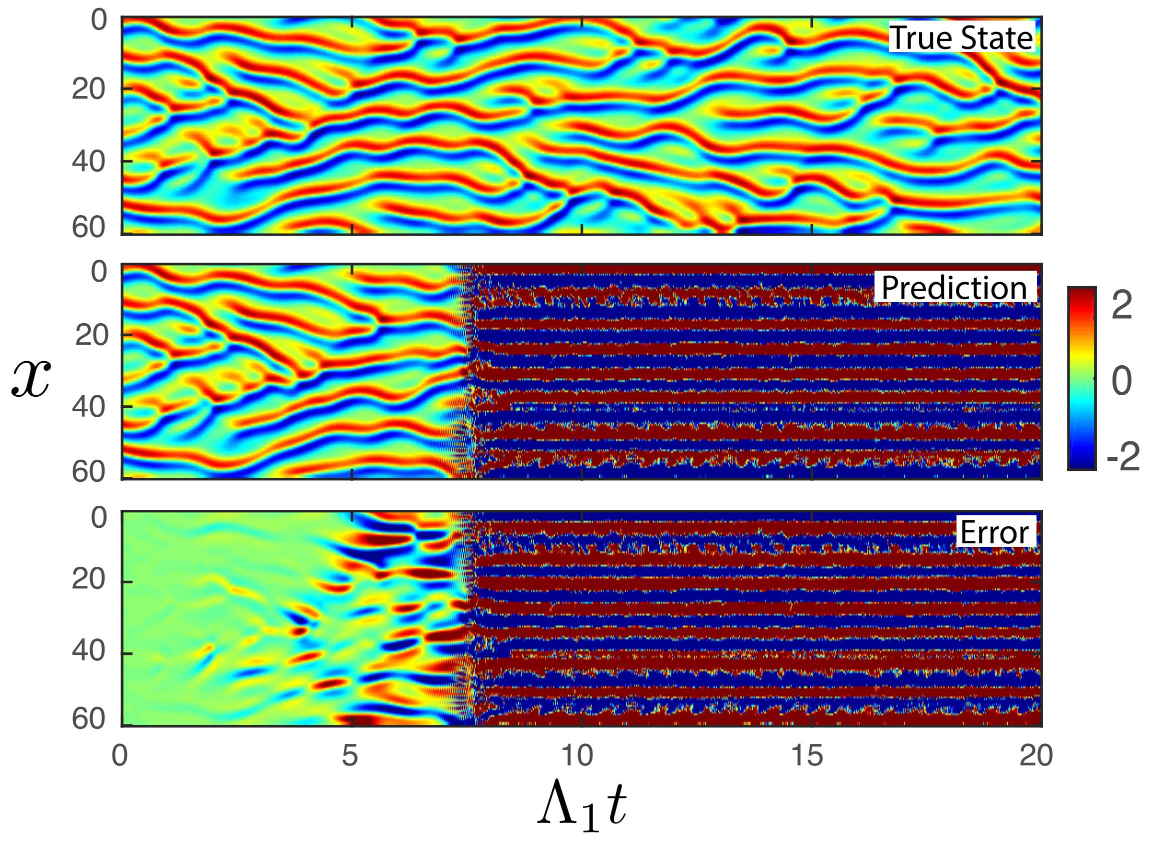


Figure 3.5: Top panel: True state, $y(x,t)$, of the standard KS system after $t = 0$. Middle panel: Reservoir prediction with a reservoir of size $D_r = 5000$ and $\rho = 3.1$. The rest of the parameters are as given in Table 3.3. Bottom panel: Difference between the reservoir prediction and the true KS state. We see that in this case, the reservoir gives us an accurate short term prediction (i.e., the 'weather') but the long term 'climate' of the autonomous reservoir dynamical system does not resemble the climate of the true KS system for this poorly chosen set of parameters. Λ_1 denotes the largest Lyapunov exponent.

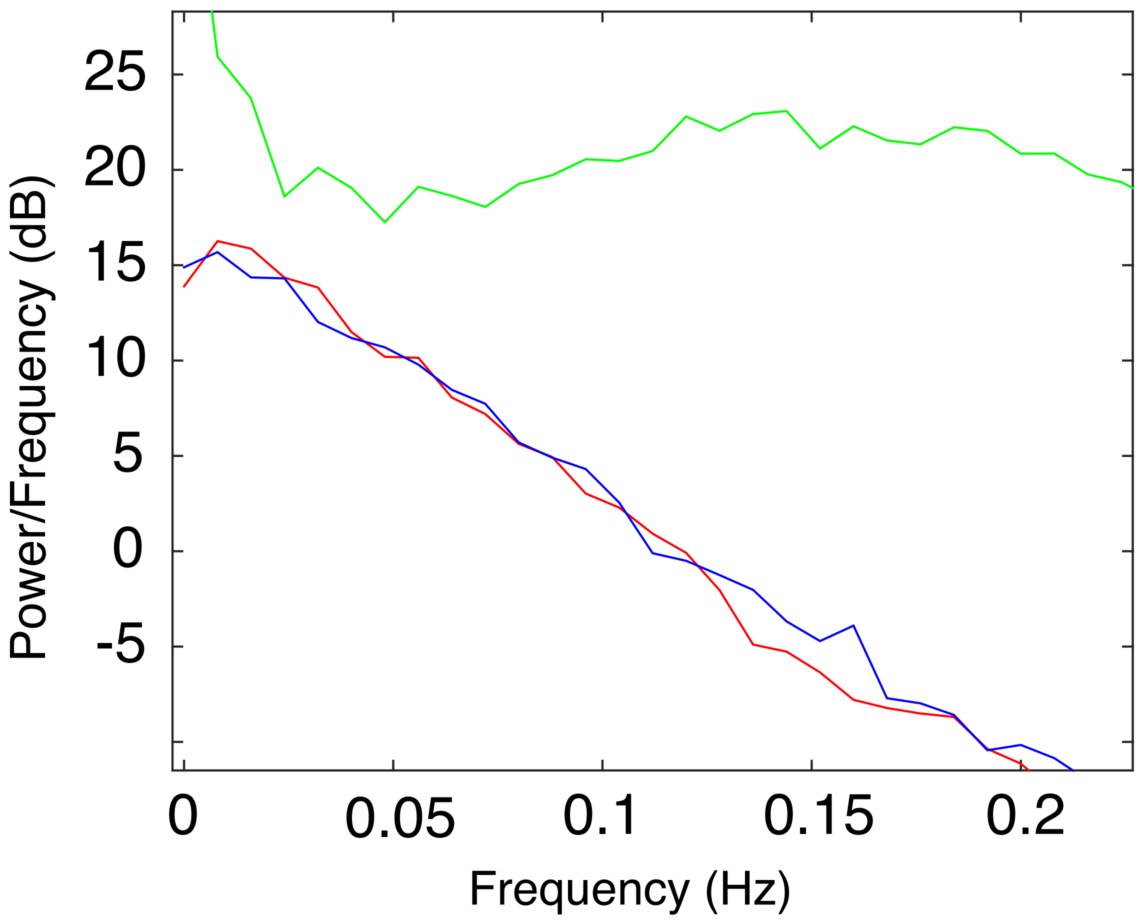


Figure 3.6: Power spectrum of the KS training data (blue), of the reservoir prediction with the same parameters as in Fig. 3.4 (red), and of the reservoir prediction with parameters as in Fig. 3.5 (green). All power spectra have been computed at a single spatial gridpoint from a time series of length 15000 Δt time steps. The power spectra are smoothed by dividing a time series into 30 intervals, computing the power spectrum of each interval and then averaging over all the intervals.

Similar to what was done in Sec. 3.2, we use our complete knowledge of the dynamics of the reservoir computer to evaluate its Lyapunov exponents. By independently evaluating the Lyapunov exponents directly from the KS equation, Eq. (3.7), we obtain the true Lyapunov exponents and compare them with the corresponding Lyapunov exponents of the reservoir dynamical system.

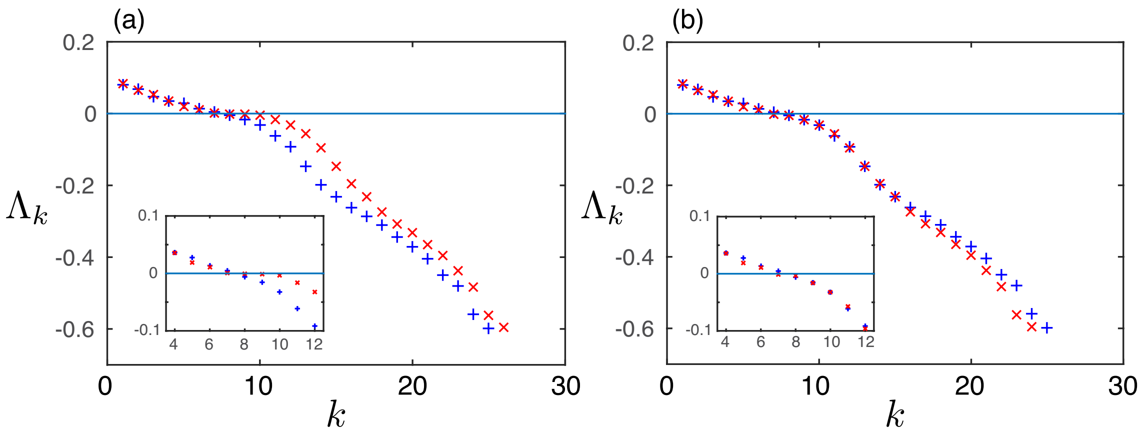


Figure 3.7: (a) Estimating the Lyapunov exponents of the homogeneous ($\mu = 0$) KS equation. First 26 Lyapunov exponents of the trained reservoir dynamical system running in autonomous prediction mode (blue '+' markers) and the standard (i.e., $\mu = 0$) KS system (red '×' markers). The parameters of Eq. (3.7) are $L = 60$, $\mu = 0$. (b) The same plot as (a), except, the two near-zero exponents of the KS system (Λ_7 and Λ_8) are removed from the spectrum. Inset: a close up of the spectra around the zero crossing. All Lyapunov exponents in this figure and Fig. 3.8 were computed from a trajectory of length 10000 Δt time steps, which we found to be sufficiently long for convergence.

3.3.1 Homogeneous KS System ($\mu = 0$)

Figure 3.7(a) shows the Lyapunov spectrum of the standard ($\mu = 0$) KS system with $L = 60$ (red '×' markers), where, by definition the subscript k is such that $\Lambda_k \geq \Lambda_{k+1}$. The Lyapunov exponents of the reservoir trained to emulate this system are shown on the same axes (blue '+' markers). We observe that the positive Lyapunov exponents of the reservoir system match the corresponding exponents of the KS system very well. However, the negative exponents of the two systems do not seem to agree with each other at first glance. We argue below that the standard KS system has three zero Lyapunov exponents, and we posit that the reservoir is unable to reproduce two of them. Indeed, Fig. 3.7(b) shows that if we remove the two of the computed exponents closest to zero (Λ_7 and Λ_8) for the KS system, the

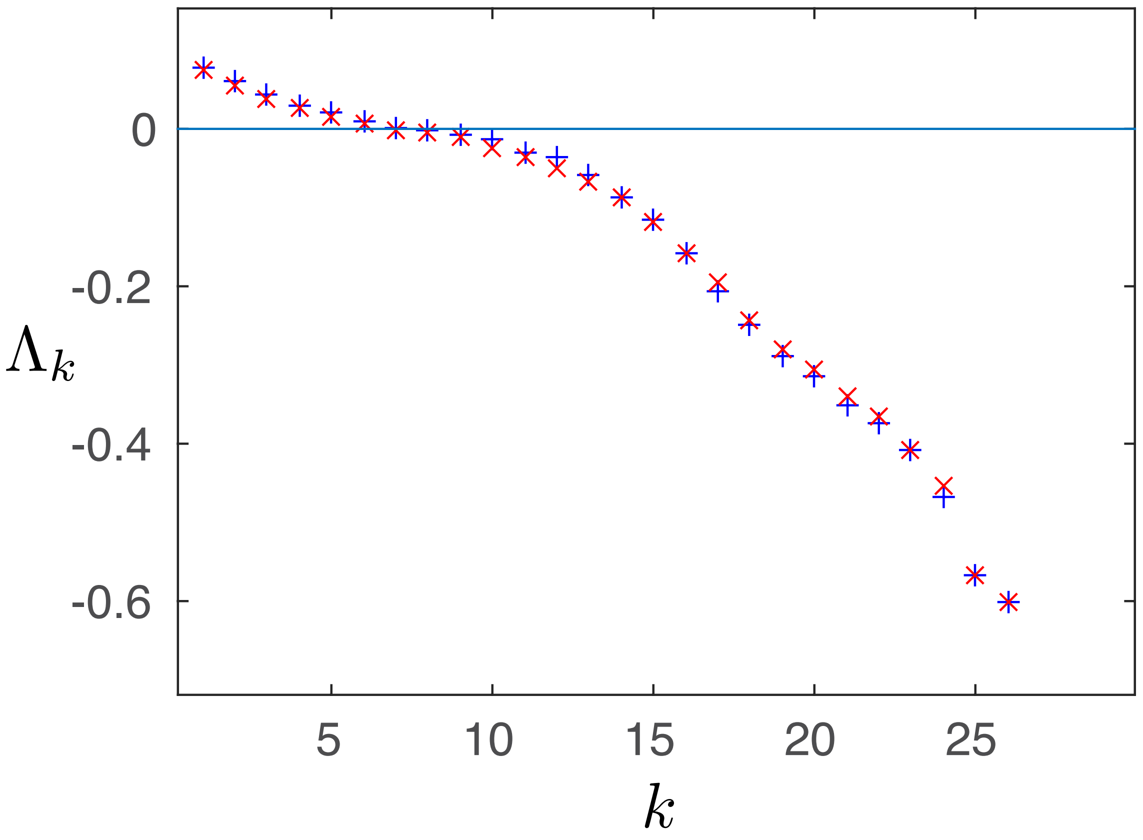


Figure 3.8: Estimating the Lyapunov exponents of the inhomogeneous ($\mu > 0$) KS equation. First 26 Lyapunov exponents of the trained reservoir dynamical system running in autonomous prediction mode (blue '+' markers) and the modified (i.e., $\mu > 0$) KS system (red '$\times$' markers). The parameters of Eq. (3.7) are $L = 60$, $\mu = 0.1$ and $\lambda = 15$.

negative Lyapunov exponents of the reservoir system match those of the KS system very well.

We show now that when $\mu = 0$ (as for Fig. 3.7), the standard KS equation (3.7) has three zero Lyapunov exponents associated with three continuous symmetries, namely time-translation invariance, space-translation invariance and the so-called Gallilean invariance. Time and space translation invariance imply that if $y(x,t)$ is a solution, then so are $y(x, t+t_0)$ and $y(x+x_0, t)$. By Gallilean invariance, we mean that for every solution $y(x,t)$ of the KS equation and an arbitrary constant v, $y(x - vt, t) + v$ is also a solution. This can be verified by direct substitution in Eq. (3.7) with $\mu = 0$. Replacing t_0, x_0, and v by differentials ($t_0 \to \delta t_0$, $x_0 \to \delta x_0$, $v \to \delta v$), we have that, $\delta y(x,t) = \frac{\partial y(x,t)}{\partial t}\delta t_0$, $\delta y(x,t) = \frac{\partial y(x,t)}{\partial x}\delta x_0$ and $\delta y(x,t) = \left[1 - t\frac{\partial y(x,t)}{\partial x}\right]\delta v$ all represent perturbations, $y(x,t) + \delta y(x,t)$, of Eq. (3.7) that are, to linear order in the differentials, solutions of Eq. (3.7). That is, all three of these $\delta y(x,t)$ are solutions of the variational equation, $\delta y_t + \delta y y_x + y \delta y_x + \delta y_{xx} + \delta y_{xxxx} = 0$. Furthermore, since the original solution $y(x,t)$ does not decay exponentially to zero, nor increase exponentially to infinity, we conclude that these three expressions for δy represent Lyapunov vectors with zero Lyapunov exponents.

To see why the reservoir does not reproduce the Gallilean symmetry-associated zero Lyapunov exponent in the $\mu = 0$ case, notice that there is a corresponding conserved quantity $c = \int y(x,t)dx$. A particular KS system trajectory in phase space is thus restricted to a hypersurface with a constant value of c (say, $c = c_0$). Since the reservoir is trained with data from a single trajectory, it does not learn the dynamics of perturbations that take the trajectory off the c_0 hypersurface. We are not certain why the reservoir does not reproduce both of the other two zero exponents.

3.3.2 Inhomogeneous KS System ($\mu = 0.1$)

As a further example that does not have additional symmetries beyond time-translation, we consider (Fig. 3.8) a KS equation with a nonzero value of μ ($L = 60, \lambda = 15, \mu = 0.1$). As before, we train the reservoir using the time series data from the symmetry broken KS equation. After training, we run the reservoir in autonomous prediction mode (Fig. 3.1(b)) and calculate its Lyapunov spectrum. Figure 3.8 shows that the reservoir reproduces the Lyapunov spectrum of the true KS system accurately in this case. Notably, in contrast with the case $\mu = 0$, this good agreement is obtained without the need of discarding two zero Lyapunov exponents. We continue to use this modified KS system in the experiments described below.

For the cases shown in Figs. 3.7(b) and 3.8, the information dimension of the attractor, as computed from the Kaplan-Yorke conjecture [[30]], is about $D_{KY} \approx 15$ (roughly, the value of k at which $\sum_{j=1}^{k} \Lambda_j$ first becomes negative). We see from Fig. 3.7(b) and Fig. 3.8 that the reservoir continues to give reasonable estimates of Λ_k even for $k > D_{KY}$. This was somewhat surprising to us, especially in view of the inaccurate reservoir estimate of Λ_3 in Sec. 3.2.

3.3.3 Effect of Measurement Noise

We now consider the effect of additive measurement noise on our Lyapunov exponent calculation scheme. We simulate measurement noise by adding a random vector $\mathbf{n}(t)$ to the training data set $\mathbf{u}(t)$ for all values of t. That is, at every time step Δt, we replace $\mathbf{u}$ in Eq. (3.1) by $\mathbf{u} + \mathbf{n}$, and we replace $\mathbf{v}_d = \mathbf{u}$ used in Eq. (3.3) by $\mathbf{v}_d = \mathbf{u} + \mathbf{n}$. The scalar elements $n_j(t)$ of the vector $\mathbf{n}(t)$, for each value of j and t, are independent, identically distributed uniform random variables in the interval $[-\alpha, \alpha]$. The constant α is chosen so that the RMS value of the noise is f times the RMS value of the noise-free signal $\mathbf{u}(t)$. Figure 3.9(a) shows the noise-free time series at a single grid point, while Figs. 3.9(b) and 3.9(c) show the same time series with added noise of strength $f = 0.05$ and $f = 0.2$, respectively. We calculate the Lyapunov exponents of the reservoir as described above. Figure 3.10 shows the Lyapunov spectrum when the noise level f is varied from 0.05 to 0.20 along with the true Lyapunov spectrum of the KS equation. We see that the reservoir results for the positive Lyapunov exponents are quite robust to noise for $f \leq 0.2$, but that the negative exponents are increasingly depressed to more negative values as f increases.

3.3.4 Effect of Training Data Length

We find that the amount of data used to train the reservoir computer can significantly affect the accuracy of the Lyapunov spectrum. The negative Lyapunov exponents are more sensitive than the positive exponents to errors due to insufficient training data. Figure 3.11 demonstrates this result through a plot of the Lyapunov

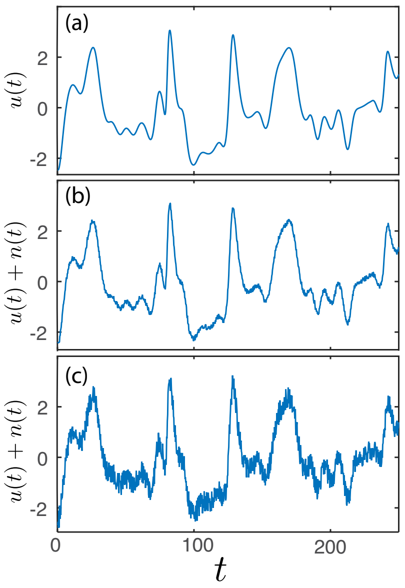


Figure 3.9: (a) Single scalar component $u(t)$ of the time series $\mathbf{u}(t)$ generated from the KS system (Eq. (3.7)) with $L = 60$, $\lambda = 15$ and $\mu = 0.1$. The time series in (a) with added noise, $u(t) + n(t)$, of noise strengths $f = 0.05$ and $f = 0.2$ are shown in (b) and (c) respectively.

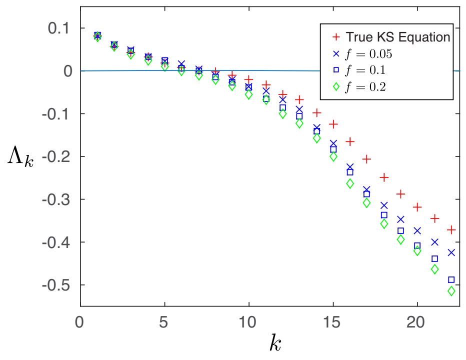


Figure 3.10: Lyapunov exponents of the reservoir trained on noisy data from the KS system ($L = 60$, $\lambda = 15$, $\mu = 0.1$). The strength of the noise added to the training data is indicated in the legend.

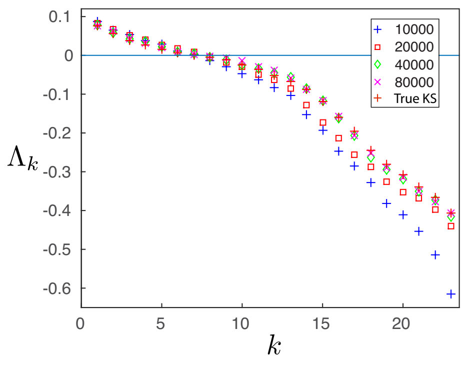


Figure 3.11: The Lyapunov spectrum of the reservoir trained using varying lengths of training data from Eq. (3.7) with parameters $L = 60$, $\lambda = 15$ and $\mu = 0.1$. The legend indicates the length of the training time series in number of Δt steps (i.e., $T/\Delta t$). For a comparison with a natural time scale of the KS system, we note that 10000 Δt time steps equals approximately 200 Lyapunov times.

spectrum of the reservoir trained on varying lengths of data from Eq. (3.7) with parameters $L = 60$, $\lambda = 15$ and $\mu = 0.1$. In this example we find that we need a training time series of greater than 20000 time steps in order to obtain a reasonably accurate estimate of the negative Lyapunov exponents (20000 time steps equals about 400 multiples of the Lyapunov time (Λ_1^{-1}) which can be considered to be a natural time scale of the KS system).

3.4 Discussion and Conclusion

We conclude that a suitably trained reservoir computing system is capable of approximating the ergodic properties of the true system that it was trained on. In the case of the Lorenz equations (Sec. 3.2), our method is successful in calculating the positive and zero Lyapunov exponents with good accuracy. The negative Lyapunov exponent of the true Lorenz system has a high magnitude, and our method is not as successful in accurately calculating the numerical value of this exponent, although it does successfully capture that its magnitude is substantially larger than that of the positive exponent. Remarkably, as shown in Sec. 3.3 for the Kuramoto-Sivashinsky system, it is possible to use the trained reservoir to calculate a large number of positive and negative Lyapunov exponents of a high dimensional spatio-temporal chaotic system with good accuracy.

In Fig. 3.11 we demonstrated that we can reproduce the Lyapunov exponents of an approximately 15-dimensional attractor from a “training” time series of 40000 points ($T/\Delta t = 40000$). By contrast, delay coordinate embedding methods that approximate the system Jacobian from nearest neighbors have been argued to require a time series of length $10^{\mathcal{D}}$ or longer [56, 57] (where $\mathcal{D}$ is the attractor dimension).

From a more general point of view, our chapter suggests that the development of machine learning techniques for model-free analysis of measured data from chaotic systems may be a fruitful subject for further research.

Chapter 4: Model-Assisted Prediction of Chaotic Dynamical Systems

This chapter is based on the paper, 'Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model', Chaos 28, 041101 (2018); https://doi.org/10.1063/1.5028373 by Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R. Hunt, Michelle Girvan, and Edward Ott ©by the American Institute of Physics

Prediction of dynamical system states (e.g., as in weather forecasting) is a common and essential task with many applications in science and technology. This task is often carried out via a system of dynamical equations derived to model the process to be predicted. Due to deficiencies in knowledge or computational capacity, application of these models will generally be imperfect and may give unacceptably inaccurate results. On the other hand data-driven methods, independent of derived knowledge of the system, can be computationally intensive and require unreasonably large amounts of data. In this chapter we consider a particular hybridization technique for combining these two approaches. Our tests of this hybrid technique suggest that it can be extremely effective and widely applicable.

4.1 Introduction

One approach to forecasting the state of a dynamical system starts by using whatever knowledge and understanding is available about the mechanisms governing the dynamics to construct a mathematical model of the system. Following that, one can use measurements of the system state to estimate initial conditions for the model that can then be integrated forward in time to produce forecasts. We refer to this approach as knowledge-based prediction. The accuracy of knowledge-based prediction is limited by any errors in the mathematical model. Another approach that has recently proven effective is to use machine learning to construct a predictor purely from extensive past measurements of the system state evolution (training data). Because the latter approach typically makes little or no use of mechanistic understanding, the amount of training data and the computational resources required can be prohibitive, especially when the system to be predicted is large and complex. The purpose of this chapter is to propose, describe, and test a general framework for combining a knowledge-based approach with a machine learning approach to build a hybrid prediction scheme with significantly enhanced potential for performance and feasibility of implementation as compared to either an approximate knowledge-based model acting alone or a machine learning model acting alone. The results of our tests of our proposed hybrid scheme suggest that it can have wide applicability for forecasting in many areas of science and technology.

Another view motivating our hybrid approach is that, when trying to predict the evolution of a system, one might intuitively expect the best results when making appropriate use of *all* the available information about the system. Here we think of both the (perhaps imperfect) knowledge-based model and the past measurement data as being two types of system information which we wish to simultaneously and efficiently utilize. Our hybrid technique combines the information in a way that does not presume either type to have greater predictive power. We note that hybrids of machine learning with other approaches have previously been applied to a variety of other tasks, but here we consider the general problem of forecasting a dynamical system with an imperfect knowledge-based model, the form of whose imperfections is unknown. Examples of such other tasks addressed by machine learning hybrids include network anomaly detection [58], credit rating [59], and chemical process modeling [60], among others. We also note a recent preprint [1] that employs a hybrid dynamical state forecasting scheme that is related to the one we present here.

To illustrate the hybrid scheme, we focus on a particular type of machine learning known as 'reservoir computing' [15,16,61], which has been previously applied to the prediction of low dimensional systems [11] and, more recently, to the prediction of large spatiotemporal chaotic systems [31,62]. We emphasize that, while our illustration is for reservoir computing with a reservoir based on an artificial neural network, we view the results as a general test of the hybrid approach. As such, these results should be relevant to other versions of machine learning [18] (such as Long Short-Term Memory networks [21]), as well as reservoir computers in which the reservoir

[1]Z. Y. Wan, P. R. Vlachas, P. Koumoutsakos, and T. P. Sapsis, Data-assisted reduced-order modeling of extreme events in complex dynamical systems, arXiv preprint arXiv:1803.03365v1 (2018). This work employs Long Short-Term Memory networks [21] rather than reservoir computing.

is implemented by various physical means (e.g., electro-optical schemes [63–65] or Field Programmable Gate Arrays [28]). A particularly dramatic example illustrating the effectiveness of the hybrid approach is shown in Figs. 4.7(d,e,f) in which, when acting alone, both the knowledge-based predictor and the reservoir machine learning predictor give fairly worthless results (prediction time of only a fraction of a Lyapunov time), but, when the same two systems are combined in the hybrid scheme, good predictions are obtained for a substantial duration of about 4 Lyapunov times. (By a 'Lyapunov time' we mean the typical time required for an e-fold increase of the distance between two initially close chaotic orbits; see Secs. 4.4 and 4.5.)

The rest of this chapter is organized as follows. In Sec. 4.2, we provide an overview of our methods for prediction by using a knowledge-based model and for prediction by exclusively using a reservoir computing approach (henceforth referred to as the reservoir-only predictor). We then describe the hybrid scheme that combines the knowledge-based model with the reservoir-only predictor. In Sec. 4.3, we describe our specific implementation of the reservoir computing scheme and the proposed hybrid scheme using a recurrent-neural-network implementation of the reservoir computer. In Secs. 4.4 and 4.5, we demonstrate our hybrid prediction approach using two examples, namely, the low-dimensional Lorenz system [66] and the high dimensional, spatiotemporal chaotic Kuramoto-Sivashinsky system [67, 68].

4.2 Prediction Methods

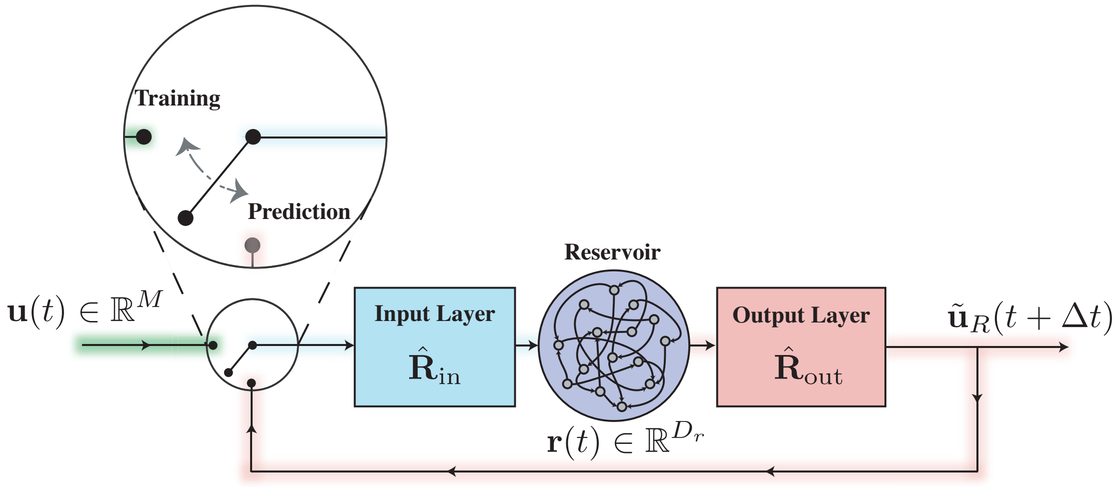

Training
Prediction
$\mathbf{u}(t) \in \mathbb{R}^M$
Input Layer
$\hat{\mathbf{R}}_{\text{in}}$
Reservoir
$\mathbf{r}(t) \in \mathbb{R}^{D_r}$
Output Layer
$\hat{\mathbf{R}}_{\text{out}}$
$\tilde{\mathbf{u}}_R(t + \Delta t)$

Figure 4.1: Schematic diagram of reservoir-only prediction setup.

We consider a dynamical system for which there is available a time series of a set of M measurable state-dependent quantities, which we represent as an M dimensional vector $\mathbf{u}(t)$. As discussed earlier, we propose a hybrid scheme to make predictions of the dynamics of the system by combining an approximate knowledge-based prediction via an approximate model with a purely data-driven prediction scheme that uses machine learning. We will compare predictions made using our hybrid scheme with the predictions of the approximate knowledge-based model alone and predictions made by exclusively using the reservoir computing model.

4.2.1 Knowledge-Based Model

We obtain predictions from the approximate knowledge-based model acting alone assuming that the knowledge-based model is capable of forecasting $\mathbf{u}(t)$ for $t > 0$ based on an initial condition $\mathbf{u}(0)$ and possibly recent values of $\mathbf{u}(t)$ for $t < 0$. For notational use in our hybrid scheme (Sec. 4.2.3), we denote integration of the knowledge-based model forward in time by a time duration Δt as,

$$\mathbf{u}_{\mathcal{K}}(t + \Delta t) = \mathcal{K}\left[\mathbf{u}(t)\right] \approx \mathbf{u}(t + \Delta t). \tag{4.1}$$

We emphasize that the knowledge-based one-step-ahead predictor $\mathcal{K}$ is imperfect and may have substantial unwanted error. In our test examples in Secs. 4.4 and 4.5 we consider prediction of continuous-time systems and take the prediction system time step Δt to be small compared to the typical time scale over which the continuous-time system changes. We note that while a single prediction time step (Δt) is small, we are interested in predicting for a large number of time steps.

4.2.2 Reservoir-Only Predictor

For the machine learning approach, we assume the knowledge of $\mathbf{u}(t)$ for times t from $-T$ to 0. This data will be used to train the machine learning model for the purpose of making predictions of $\mathbf{u}(t)$ for $t > 0$. In particular we use a reservoir computer, described as follows.

A reservoir computer (Fig. 4.1) is constructed with an artificial high dimensional dynamical system, known as the reservoir whose state is represented by the D_r dimensional vector $\mathbf{r}(t)$, $D_r \gg M$. We note that ideally the forecasting accuracy of a reservoir-only prediction increases with D_r, but that D_r is typically limited by computational cost considerations. The reservoir is coupled to an input through an Input-to-Reservoir coupling $\hat{\mathbf{R}}_{in}\left[\mathbf{u}(t)\right]$ which maps the M-dimensional input vector, $\mathbf{u}$, at time t, to each of the D_r reservoir state variables. The output is defined through a Reservoir-to-Output coupling $\hat{\mathbf{R}}_{out}\left[\mathbf{r}(t), \mathbf{p}\right]$, where $\mathbf{p}$ is a large set of *adjustable* parameters. In the task of prediction of state variables of dynamical systems the reservoir computer is used in two different configurations. One of the configurations we call the 'training' phase, and the other one we called the 'prediction' phase. In the training phase, the reservoir system is configured according to Fig. 4.1 with the switch in the position labeled 'Training'. In this phase, the reservoir state evolves from $t = -T$ to $t = 0$ according to the equation,

$$\mathbf{r}(t + \Delta t) = \hat{\mathbf{G}}_R\left[\hat{\mathbf{R}}_{in}\left[\mathbf{u}(t)\right], \mathbf{r}(t)\right], \tag{4.2}$$

where the nonlinear function $\hat{\mathbf{G}}_R$ and the (usually linear) function $\hat{\mathbf{R}}_{in}$ depend on the choice of the reservoir implementation. Next, we make a particular choice of the parameters $\mathbf{p}$ such that the output function $\hat{\mathbf{R}}_{out}\left[\mathbf{r}(t),\mathbf{p}\right]$ satisfies,

$$\hat{\mathbf{R}}_{out}\left[\mathbf{r}(t),\mathbf{p}\right] \approx \mathbf{u}(t), \tag{4.3}$$

for $-T < t \leq 0$. We achieve this by minimizing the error between $\tilde{\mathbf{u}}_R(t) = \hat{\mathbf{R}}_{out}\left[\mathbf{r}(t),\mathbf{p}\right]$ and $\mathbf{u}(t)$ for $-T < t \leq 0$ using a suitable error metric and optimization algorithm on the adjustable parameter vector $\mathbf{p}$.

In the prediction phase, for $t \geq 0$, the switch is placed in position labeled 'Prediction' indicated in Fig. 4.1. The reservoir state now evolves autonomously with a feedback loop according to the equation,

$$\mathbf{r}(t+\Delta t) = \hat{\mathbf{G}}_R\left[\hat{\mathbf{R}}_{in}\left[\tilde{\mathbf{u}}_R(t)\right],\mathbf{r}(t)\right], \tag{4.4}$$

where, $\tilde{\mathbf{u}}_R(t) = \hat{\mathbf{R}}_{out}\left[\mathbf{r}(t),\mathbf{p}\right]$ is taken as the prediction from this reservoir-only approach. It has been shown [11] that this procedure can successfully generate a time series $\tilde{\mathbf{u}}_R(t)$ that approximates the true state $\mathbf{u}(t)$ for $t > 0$. Thus $\tilde{\mathbf{u}}_R(t)$ is our reservoir-based prediction of the evolution of $\mathbf{u}(t)$. If, as assumed henceforth, the dynamical system being predicted is chaotic, the exponential divergence of initial conditions in the dynamical system implies that any prediction scheme will only be able to yield an accurate prediction for a limited amount of time.

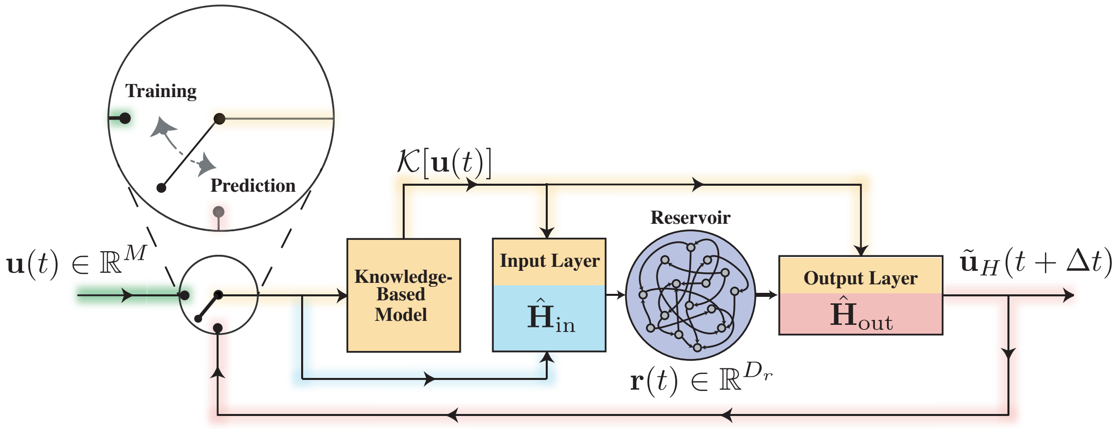

Training
Prediction
$\mathcal{K}[\mathbf{u}(t)]$
Reservoir
$\mathbf{u}(t) \in \mathbb{R}^M$
Knowledge-Based Model
Input Layer
$\hat{\mathbf{H}}_{\text{in}}$
Output Layer
$\hat{\mathbf{H}}_{\text{out}}$
$\tilde{\mathbf{u}}_H(t+\Delta t)$
$\mathbf{r}(t) \in \mathbb{R}^{D_r}$

Figure 4.2: Schematic diagram of the hybrid prediction setup.

4.2.3 Hybrid Scheme

The hybrid approach we propose combines both the knowledge-based model and the reservoir-only predictor. Our hybrid approach is outlined in the schematic diagram shown in Fig. 4.2.

As in the reservoir-only approach, the hybrid scheme has two phases, the training phase and the prediction phase. In the training phase (with the switch in position labeled ‘Training’ in Fig. 4.2), the training data $\mathbf{u}(t)$ from $t = -T$ to $t = 0$ is fed into both the knowledge-based predictor and the reservoir. At each time t, the output of the knowledge-based predictor is the one-step ahead prediction $\mathcal{K}\left[\mathbf{u}(t)\right]$. The reservoir evolves according to the equation

$$\mathbf{r}(t+\Delta t) = \hat{\mathbf{G}}_H\left[\mathbf{r}(t), \hat{\mathbf{H}}_{in}\left[\mathcal{K}\left[\mathbf{u}(t)\right], \mathbf{u}(t)\right]\right] \tag{4.5}$$

for $-T \leq t \leq 0$, where the (usually linear) function $\hat{\mathbf{H}}_{in}$ couples the reservoir network with the inputs to the reservoir, in this case $\mathbf{u}(t)$ and $\mathcal{K}\left[\mathbf{u}(t)\right]$. As earlier, we modify a set of adjustable parameters $\mathbf{p}$ in a predefined output function so that

$$\hat{\mathbf{H}}_{out}\left[\mathcal{K}\left[\mathbf{u}(t-\Delta t)\right], \mathbf{r}(t), \mathbf{p}\right] \approx \mathbf{u}(t) \tag{4.6}$$

for $-T < t \leq 0$, which is achieved by minimizing the error between the right-hand side and the left-hand side of Eq. (4.6), as discussed earlier (Sec. 4.2.2) for the reservoir-only approach. Note that both the knowledge-based model and the reservoir feed into the output layer (Eq. (4.6) and Fig. 4.2) so that the training can be thought of as optimally deciding on how to weight the information from the knowledge-based and reservoir components.

For the prediction phase (the switch is placed in the position labeled ‘Prediction’ in Fig. 4.2) the feedback loop is closed allowing the system to evolve autonomously. The dynamics of the system will then be given by

$$\mathbf{r}(t+\Delta t) = \hat{\mathbf{G}}_H \left[\mathbf{r}(t), \hat{\mathbf{H}}_{in} \left[\mathcal{K}\left[\tilde{\mathbf{u}}_H(t)\right], \tilde{\mathbf{u}}_H(t)\right]\right], \tag{4.7}$$

where $\tilde{\mathbf{u}}_H(t) = \hat{\mathbf{H}}_{out}\left[\mathcal{K}\left[\tilde{\mathbf{u}}_H(t-\Delta t)\right], \mathbf{r}(t), \mathbf{p}\right]$, is the prediction of the prediction of the hybrid system.

4.3 Implementation

In this section we provide details of our specific implementation and discuss the prediction performance metrics we use to assess and compare the various prediction schemes. Our implementation of the reservoir computer closely follows Ref. [11]. Note that, in the reservoir training, no knowledge of the dynamics and details of the reservoir (the network within the circles in Figs. 4.1 and 4.2) is used (this contrasts with other machine learning techniques [18]): only the $-T \leq t \leq 0$ training data is used ($\mathbf{u}(t)$, $\mathbf{r}(t)$, and, in the case of the hybrid, $\mathcal{K}[\mathbf{u}(t)]$). This feature implies that reservoir computers, as well as the reservoir-based hybrid are insensitive to the specific reservoir implementation. In this chapter, our illustrative implementation of the reservoir computer uses an artificial neural network for the realization of the reservoir. We mention, however, that alternative implementation strategies such as utilizing nonlinear optical devices [63–65] and Field Programmable Gate Arrays [28] can also be used to construct the reservoir component of our hybrid scheme (Fig. 4.2) and offer potential advantages, particularly with respect to speed.

4.3.1 Reservoir-Only and Hybrid Implementations

Here we consider that the high-dimensional reservoir is implemented by a large, weighted, directed, low degree Erdős-Rènyi network of D_r nonlinear, neuron-like units in which the network is described by an adjacency matrix $\mathbf{A}$ (we stress that the following implementations are somewhat arbitrary, and are intended as illustrating typical results that might be expected). The network is constructed to have an average degree denoted by $\langle d \rangle$, and the nonzero elements of $\mathbf{A}$, representing the edge weights in the network, are initially chosen independently from the uniform distribution over the interval $[-1, 1]$. All the edge weights in the network are then uniformly scaled via multiplication of the adjacency matrix by a constant factor to set the largest magnitude eigenvalue of the matrix to a quantity ρ, which is called the 'spectral radius' of $\mathbf{A}$. The state of the reservoir, given by the vector $\mathbf{r}(t)$, consists of the components r_j for $1 \leq j \leq D_r$ where $r_j(t)$ denotes the scalar state of the j^{th} node in the network. When evaluating prediction based purely on a reservoir system alone, the reservoir is coupled to the M dimensional input through a $D_r \times M$ dimensional matrix $\mathbf{W}_{in}$, such that in Eq. (4.2) $\hat{\mathbf{R}}_{in}\left[\mathbf{u}(t)\right] = \mathbf{W}_{in}\mathbf{u}(t)$, and each row of the matrix $\mathbf{W}_{in}$ has exactly one randomly chosen nonzero element. Each nonzero element of the matrix is independently chosen from the uniform distribution on the interval $[-\sigma, \sigma]$. We choose the hyperbolic tangent function for the form of the nonlinearity at the nodes, so that the specific training phase equation for our reservoir setup corresponding to Eq. (4.2) is

$$\mathbf{r}(t+\Delta t) = \tanh\left[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)\right], \tag{4.8}$$

where the hyperbolic tangent applied on a vector is defined as the vector whose components are the hyperbolic tangent function acting on each element of the argument vector individually.

We choose the form of the output function to be $\hat{\mathbf{R}}_{out}(\mathbf{r},\mathbf{p}) = \mathbf{W}_{out}\mathbf{r}^{\star}$, in which the output parameters (previously symbolically represented by $\mathbf{p}$) will henceforth be take to be the elements of the matrix $\mathbf{W}_{out}$, and the vector $\mathbf{r}^{\star}$ is defined such that $r_j^{\star}$ equals r_j for odd j, and equals r_j^2 for even j (it was empirically found that this choice of $\mathbf{r}^{\star}$ works well for our examples in both Sec. 4.4 and Sec. 4.5, see also [22, 31]). We run the reservoir for $-T \leq t \leq 0$ with the switch in Fig. 4.1 in the 'Training' position. We then use Tikhonov regularized linear regression [32] to train $\tilde{\mathbf{u}}_R(t) = \mathbf{W}_{out}\mathbf{r}^{\star}(t)$ to approximate $\mathbf{u}(t)$. That is, we minimize the quantity $\sum_{m=1}^{T/\Delta t} \| \mathbf{u}(-m\Delta t) - \tilde{\mathbf{u}}_R(-m\Delta t)\|^2 + \beta\|\mathbf{W}_{out}\|^2$, where $\|\mathbf{W}_{out}\|^2$ is the sum of the squares of the matrix elements of $\mathbf{W}_{out}$ and the regularization parameter β is a small positive number introduced to avoid overfitting. Since $\tilde{\mathbf{u}}_R$ depends linearly on the elements of $\mathbf{W}_{out}$, this minimization is a standard linear regression problem.

Once the output parameters (the matrix elements of $\mathbf{W}_{out}$) are determined, we run the system in the configuration depicted in Fig. 4.1 with the switch in the

‘Prediction’ position according to the equations,

$$\tilde{\mathbf{u}}_R(t) = \mathbf{W}_{out}\mathbf{r}^\star(t)\mathbf{r}(t+\Delta t) = \tanh\left[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\tilde{\mathbf{u}}_R(t)\right], \tag{4.9}$$

corresponding to Eq. (4.4). Here $\tilde{\mathbf{u}}_R(t)$ denotes the prediction of $\mathbf{u}(t)$ for $t > 0$ made by the reservoir-only model.

Next, we describe the implementation of the hybrid prediction scheme. The reservoir component of our hybrid scheme is implemented in the same fashion as in the reservoir-only model given above. In the training phase for $-T < t \leq 0$, when the switch in Fig. 4.2 is in the ‘Training’ position, the specific form of Eq. (4.5) used is given by

$$\mathbf{r}(t+\Delta t) = \tanh\left[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\begin{pmatrix}\mathcal{K}\left[\mathbf{u}(t)\right] \\ \mathbf{u}(t)\end{pmatrix}\right] \tag{4.10}$$

As earlier, we choose the matrix $\mathbf{W}_{in}$ (which is now $D_r \times (2M)$ dimensional) to have exactly one nonzero element in each row. The nonzero elements are independently chosen from the uniform distribution on the interval $[-\sigma, \sigma]$. Each nonzero element can be interpreted to correspond to a connection to a particular reservoir node. These nonzero elements are randomly chosen such that a fraction γ of the reservoir nodes are connected exclusively to the raw input $\mathbf{u}(t)$ and the remaining fraction are connected exclusively to the the output of the model based predictor $\mathcal{K}[\mathbf{u}(t)]$.

Similar to the reservoir-only case, we choose the form of the output function to be

$$\hat{\mathbf{H}}_{out}\left[\mathcal{K}\left[\mathbf{u}(t-\Delta t)\right],\mathbf{r}(t),\mathbf{p}\right]=\mathbf{W}_{out}\begin{pmatrix}\mathcal{K}\left[\mathbf{u}(t-\Delta t)\right]\\ \\ \mathbf{r}^{\star}(t)\end{pmatrix}, \tag{4.11}$$

Where, as in the reservoir-only case, $\mathbf{W}_{out}$ now plays the role of $\mathbf{p}$. Again, as in the reservoir-only case, $\mathbf{W}_{out}$ is determined via Tikhonov regularized regression.

In the prediction phase for $t > 0$, when the switch in Fig. 4.2 is in position labeled 'Prediction', the input $\mathbf{u}(t)$ is replaced by the output at the previous time step and the equation analogous to Eq. (4.7) is given by,

$$\tilde{\mathbf{u}}_H(t)=\mathbf{W}_{out}\begin{pmatrix}\mathcal{K}\left[\mathbf{u}(t)\right]\\ \\ \mathbf{r}^{\star}(t)\end{pmatrix},\mathbf{r}(t+\Delta t)=\tanh\left[\mathbf{A}\mathbf{r}(t)+\mathbf{W}_{in}\begin{pmatrix}\mathcal{K}\left[\tilde{\mathbf{u}}_H\right]\\ \\ \tilde{\mathbf{u}}_H\end{pmatrix}\right]. \tag{4.12}$$

The vector time series $\tilde{\mathbf{u}}_H(t)$ denotes the prediction of $\mathbf{u}(t)$ for $t > 0$ made by our hybrid scheme.

4.3.2 Training Reusability

In the prediction phase, $t > 0$, chaos combined with a small initial condition error, $\|\tilde{\mathbf{u}}(0) - \mathbf{u}(0)\| \ll \|\mathbf{u}(0)\|$, and imperfect reproduction of the true system dynamics by the prediction method lead to a roughly exponential increase of the prediction error $\|\tilde{\mathbf{u}}(t) - \mathbf{u}(t)\|$ as the prediction time t increases. Eventually, the prediction error becomes unacceptably large. By choosing a value for the largest acceptable prediction error, one can define a "valid time" t_v for a particular trial. As our examples in the following sections show, t_v is typically much less than the necessary duration T of the training data required for either reservoir-only prediction or for prediction by our hybrid scheme. However, it is important to point out that the reservoir and hybrid schemes have the property of *training reusability*. That is, once the output parameters $\mathbf{p}$ (or $\mathbf{W}_{out}$) are obtained using the training data in $-T \leq t \leq 0$, the same $\mathbf{p}$ can be used over and over again for subsequent predictions, without retraining $\mathbf{p}$. For example, say that we now desire another prediction starting at some later time $t_0 > 0$. In order to do this, the reservoir system (Fig. 4.1) or the hybrid system (Fig. 4.2) *with the predetermined* $\mathbf{p}$, is first run with the switch in the 'Training' position for a time, $(t_0 - \xi) < t < t_0$. This is done in order to resynchronize the reservoir state $\mathbf{r}(t_0)$ to the dynamics of the true system. Then, at time $t = t_0$, the switch (Figs. 4.1 and 4.2) is moved to the 'Prediction' position, and the output $\tilde{\mathbf{u}}_R$ or $\tilde{\mathbf{u}}_H$ is taken as the prediction for $t > t_0$. We find that with $\mathbf{p}$ predetermined, the time required for re-synchronization ξ turns out to be very small compared to t_v, which is in turn small compared to the training time T.

4.3.3 Assessments of Prediction Methods

We wish to compare the effectiveness of different prediction schemes (knowledge-based, reservoir-only, or hybrid). As previously mentioned, for each independent prediction, we quantify the duration of accurate prediction with the corresponding "valid time", denoted t_v, defined as the elapsed time before the normalized error

$E(t)$ first exceeds some value f, $0 < f < 1$, $E(t_v) = f$, where

$$E(t) = \frac{||\mathbf{u}(t) - \widetilde{\mathbf{u}}(t)||}{\langle ||\mathbf{u}(t)||^2 \rangle^{1/2}}, \tag{4.13}$$

and the symbol $\tilde{\mathbf{u}}(t)$ now stands for the prediction [either $\tilde{\mathbf{u}}_{\mathcal{K}}(t), \tilde{\mathbf{u}}_R(t)$ or $\tilde{\mathbf{u}}_H(t)$ as obtained by either of the three prediction methods (knowledge-based, reservoir-based, or hybrid)].

In what follows we use $f = 0.4$. We test all three methods on 20 disjoint time intervals of length τ in a long run of the true dynamical system. For each prediction method, we evaluate the valid time over many independent prediction trials. Further, for the reservoir-only prediction and the hybrid schemes, we use 32 different random realizations of $\mathbf{A}$ and $\mathbf{W}_{in}$, for each of which we separately determine the training output parameters $\mathbf{W}_{out}$; then we predict on each of the 20 time intervals for each such random realization, taking advantage of training reusability (Sec. 4.3.2). Thus, there are a total of 640 different trials for the reservoir-only and hybrid system methods, and 20 trials for the knowledge-based method. We use the median valid time across all such trials as a measure of the quality of prediction of the corresponding scheme, and the span between the first and third quartiles of the t_v values as a measure of variation in this metric of the prediction quality.

4.4 Lorenz system

The Lorenz system [66] is described by the equations,

$$\frac{dx}{dt} = -ax + ay, \frac{dy}{dt} = bx - y - xz, \frac{dz}{dt} = -cz + xy,$$

For our "true" dynamical system, we use $a = 10$, $b = 28$, $c = 8/3$ which we use to generate simulated data in $-T \leq t \leq 0$ and to generate true orbits in $t > 0$ for comparison with our predictions. For our knowledge-based predictor, we use an 'imperfect' version of the Lorenz equations to represent an approximate, imperfect model that might be encountered in a real life situation. Our imperfect model differs from the true Lorenz system given in Eq. (6.20) only via a change in the system parameter b in Eq. (6.20) to $b(1 + \epsilon)$. The error parameter ϵ is thus a dimensionless quantification of the discrepancy between our knowledge-based predictor and the 'true' Lorenz system. We emphasize that, although we simulate model error by a shift of the parameter b, we view this to represent a general model error of *unknown form*. For example we can view our parameter mismatch ϵ as a surrogate for a situation where the best available model differs by order ϵ from the dynamics due to factors such as imperfect knowledge basis of the system, too crude subgrid-scale modeling, etc. (e.g., see Ref. [1], where the knowledge-based system is a Galerkin approximation of a higher dimensional true system). This view is reflected by the fact that our reservoir and hybrid methods do not incorporate knowledge that the system error in our experiments results from an imperfect parameter value of a system with Lorenz form. Next, for the reservoir computing component of the hybrid scheme, we construct a network-based reservoir as discussed in Sec. 4.2.2 for various reservoir sizes D_r and with the parameters listed in Table 4.1.

Parameter	Value	Parameter	Value
ρ	0.4	T	100
$\langle d \rangle$	3	γ	0.5
σ	0.15	τ	250
Δt	0.1	ξ	10

Table 4.1: Reservoir parameters ρ, $\langle d \rangle$, σ, Δt, training time T, hybrid parameter γ, and evaluation parameters τ, ξ for the Lorenz system prediction.

Figure 4.3 shows an illustrative example of one prediction trial using the hybrid method. The horizontal axis is the time in units of the Lyapunov time λ_{max}^{-1}, where λ_{max} denotes the largest Lyapunov exponent of the system, Eqs. (6.20). The vertical dashed lines in Fig. 4.3 indicate the valid time t_v (Sec. 4.3.3) at which $E(t)$ (Eq. (4.13)) first reaches the value $f = 0.4$. The valid time determination for this example with $\epsilon = 0.05$ and $D_r = 500$ is illustrated in Fig. 4.4. Notice that we get low prediction error for about 10 Lyapunov times.

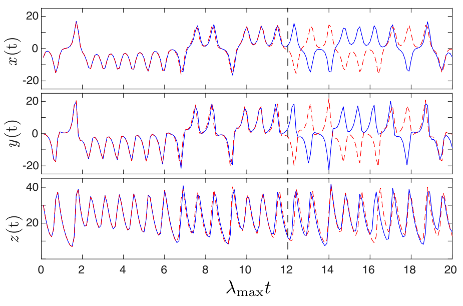


Figure 4.3: Prediction of the Lorenz system using the hybrid prediction setup. The blue line shows the true state of the Lorenz system and the red dashed line shows the prediction. Prediction begins at $t = 0$. The vertical black dashed line marks the point where this prediction is no longer considered valid by the valid time metric with $f = 0.4$. The error in the approximate model used in the knowledge-based component of the hybrid scheme is $\epsilon = 0.05$.

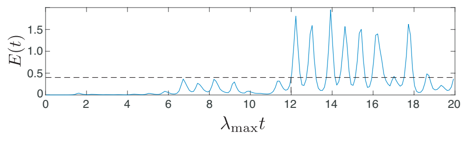


Figure 4.4: Normalized error $E(t)$ versus time of the Lorenz prediction trial shown in Fig. 4.3. The prediction error remains below the defined threshold $(E(t) < 0.4)$ for about 12 Lyapunov times.

The red upper curve in Fig. 4.5 shows the dependence on reservoir size D_r of results for the median valid time (in units of Lyapunov time, $\lambda_{max}t$, and with $f = 0.4$) of the predictions from a hybrid scheme using a reservoir system combined with our imperfect model with an error parameter of $\epsilon = 0.05$. The error bars span the first and third quartiles of our trials which are generated as described in Sec. 4.3.3. The black middle curve in Fig. 4.5 shows the corresponding results for predictions using the reservoir-only scheme. The blue lower curve in Fig. 4.5 shows the result for prediction using only the $\epsilon = 0.05$ imperfect knowledge-based model (since this result does not depend on D_r, the blue curve is horizontal and the error bars are the same at each value of D_r). Note that, even though the knowledge-based prediction alone is very bad, when used in the hybrid, it results in a large prediction improvement relative to the reservoir-only prediction. Moreover, this improvement is seen for all values of the reservoir sizes tested. Note also that the valid time for the hybrid with a reservoir size of $D_r = 50$ is comparable to the valid time for a reservoir-only scheme at $D_r = 500$. This suggests that our hybrid method can substantially reduce reservoir computational expense even with a knowledge-based model that has low predictive power on its own.

Figure. 4.6 shows the dependence of prediction performance on the model error

ϵ with the reservoir size held fixed at $D_r = 50$. For the wide range of the error ϵ we have tested, the hybrid performance is much better than either its knowledge-based component alone or reservoir-only component. Figures 4.5 and 4.6, taken together, suggest the potential robustness of the utility of the hybrid approach.

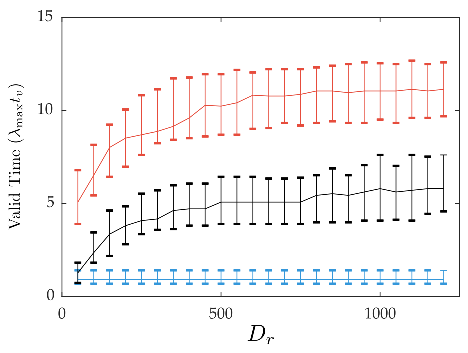


Figure 4.5: Reservoir size (D_r) dependence of the median valid time using the hybrid prediction scheme (red upper plot), the reservoir-only (black middle plot) and the knowledge-based model only methods. The model error is fixed at $\epsilon = 0.05$. Since the knowledge based model (blue) does not depend on D_r, its plot is a horizontal line. Error bars span the range between the 1st and 3rd quartiles of the trials.

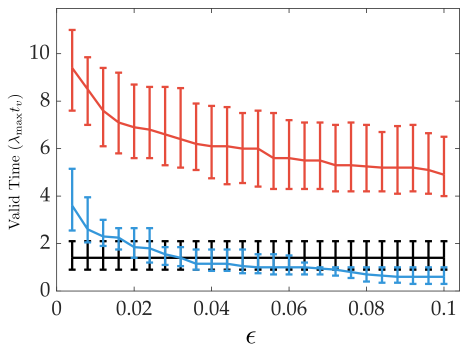


Figure 4.6: Valid times for different values of model error (ϵ) with $f = 0.4$. The reservoir size is fixed at $D_r = 50$. Plotted points represent the median and error bars span the range between the 1st and 3rd quartiles. The meaning of the colors is the same as in Fig. 4.5. Since the reservoir only scheme (black) does not depend on ϵ, its plot is a horizontal line. Similar to Fig. 4.5, the small reservoir alone cannot predict well for a long time, but the hybrid model, which combines the inaccurate knowledge-based model and the small reservoir performs well across a broad range of ϵ.

4.5 Kuramoto-Sivashinsky equations

In this example, we test how well our hybrid method, using an inaccurate knowledge-based model combined with a relatively small reservoir, can predict systems that exhibit high dimensional spatiotemporal chaos. Specifically, we use simulated data from the one-dimensional Kuramoto-Sivashinsky (KS) equation for $y(x,t)$,

$$y_t = -yy_x - y_{xx} - y_{xxxx} \tag{4.14}$$

Our simulation calculates $y(x,t)$ on a uniformly spaced grid with spatially periodic boundary conditions such that $y(x,t) = y(x+L,t)$, with a periodicity length of $L = 35$, a grid size of $Q = 64$ grid points (giving a intergrid spacing of $\Delta x = \frac{L}{Q} \approx 0.547$), and a sampling time of $\Delta t = 0.25$. For these parameters we found that the maximum Lyapunov exponent, λ_{max}, is positive ($\lambda_{max} \approx 0.07$), indicating that this system is chaotic. We define a vector of $y(x,t)$ values at each grid point as the input to each of our predictors:

$$\mathbf{u}(t) = \left[y\left(\frac{L}{Q}, t\right), y\left(\frac{2L}{Q}, t\right), \ldots, y\left(L, t\right) \right]^T. \tag{4.15}$$

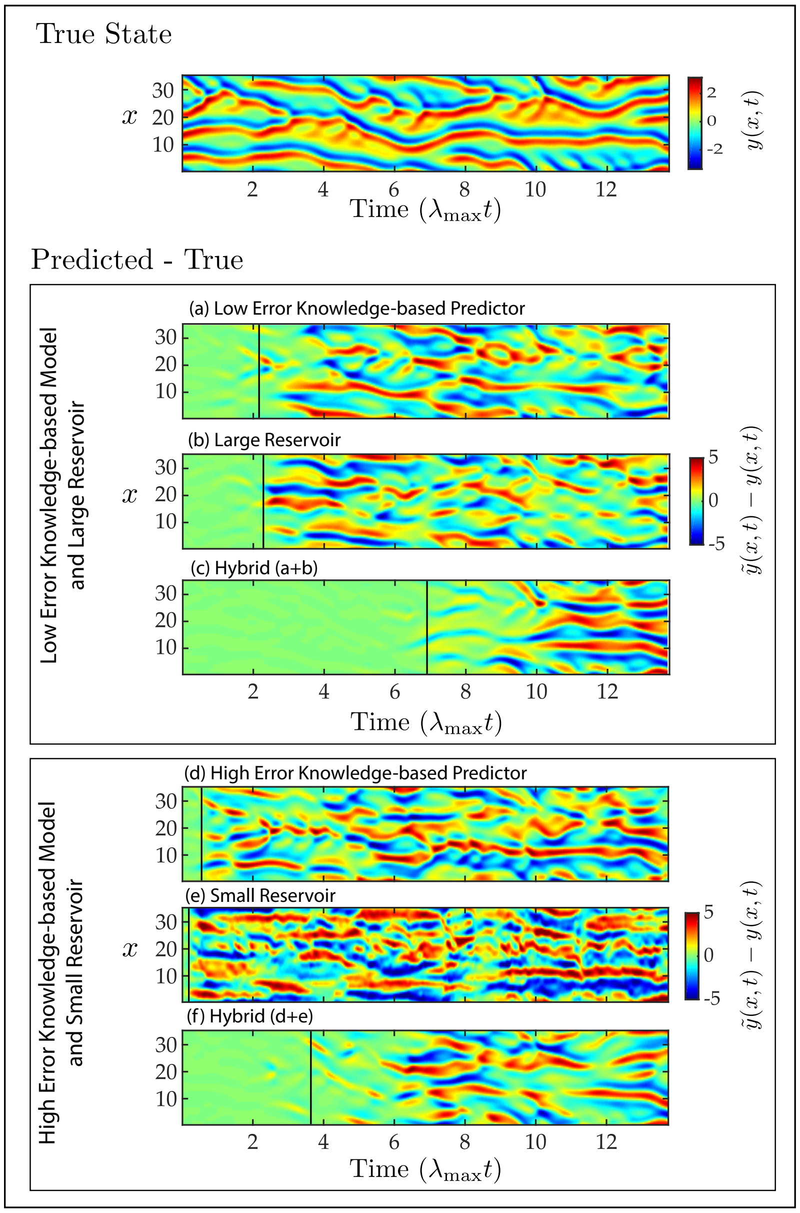


Figure 4.7: The topmost panel shows the true solution of the KS equation (Eq. (4.14)). Each of the six panels labeled (a) through (f) shows the difference between the true state of the KS system and the prediction made by a specific prediction scheme. The three panels (a), (b), and (c), respectively, show the results for a low error knowledge-based model ($\epsilon = 0.01$), a reservoir-only prediction scheme with a large reservoir ($D_r = 8000$), and the hybrid scheme composed of ($D_r = 8000$, $\epsilon = 0.01$). The three panels, (d), (e), and (f) respectively show the corresponding results for a highly imperfect knowledge-based model ($\epsilon = 0.1$), a reservoir-only prediction scheme using a small reservoir ($D_r = 500$), and the hybrid scheme with ($D_r = 500$, $\epsilon = 0.1$).

For our approximate knowledge-based predictor, we use the same simulation method as the original Kuramoto-Sivashinsky equations with an error parameter ϵ added to the coefficient of the second derivative term as follows,

$$y_t = -yy_x - (1+\epsilon)y_{xx} - y_{xxxx}. \tag{4.16}$$

For sufficiently small ϵ, Eq. (4.16) corresponds to a very accurate knowledge-based model of the true KS system, which becomes less and less accurate as ϵ is increased.

Illustrations of our main result are shown in Figs. 4.7 and 4.8, where we use the parameters in Table 4.2. In the top panel of Fig. 4.7, we plot a computed solution of Eq. (4.14) which we regard as the true dynamics of a system to be predicted; the spatial coordinate $x \in [0, L]$ is plotted vertically, the time in Lyapunov units ($\lambda_{max}t$) is plotted horizontally, and the value of $y(x,t)$ is color coded with the most positive and most negative y values indicated by red and blue, respectively. Below this top panel are six panels labeled (a-f) in which the color coded quantity is the prediction error $\tilde{y}(x,t) - y(x,t)$ of different predictions $\tilde{y}(x,t)$. In panels (a), (b) and (c), we consider a case ($\epsilon = 0.01$, $D_r = 8000$) where both the knowledge-based model (panel (a)) and the reservoir-only predictor (panel (b)) are fairly accurate; panel (c) shows the hybrid prediction error. In panels (d), (e), and (f), we consider a different case ($\epsilon = 0.1$, $D_r = 500$) where both the knowledge-based model (panel (d)) and the reservoir-only predictor (panel (e)) are relatively inaccurate; panel (f) shows the hybrid prediction error. In our color coding, low prediction error corresponds to the green color. The vertical solid lines denote the valid times for this run with $f = 0.4$.

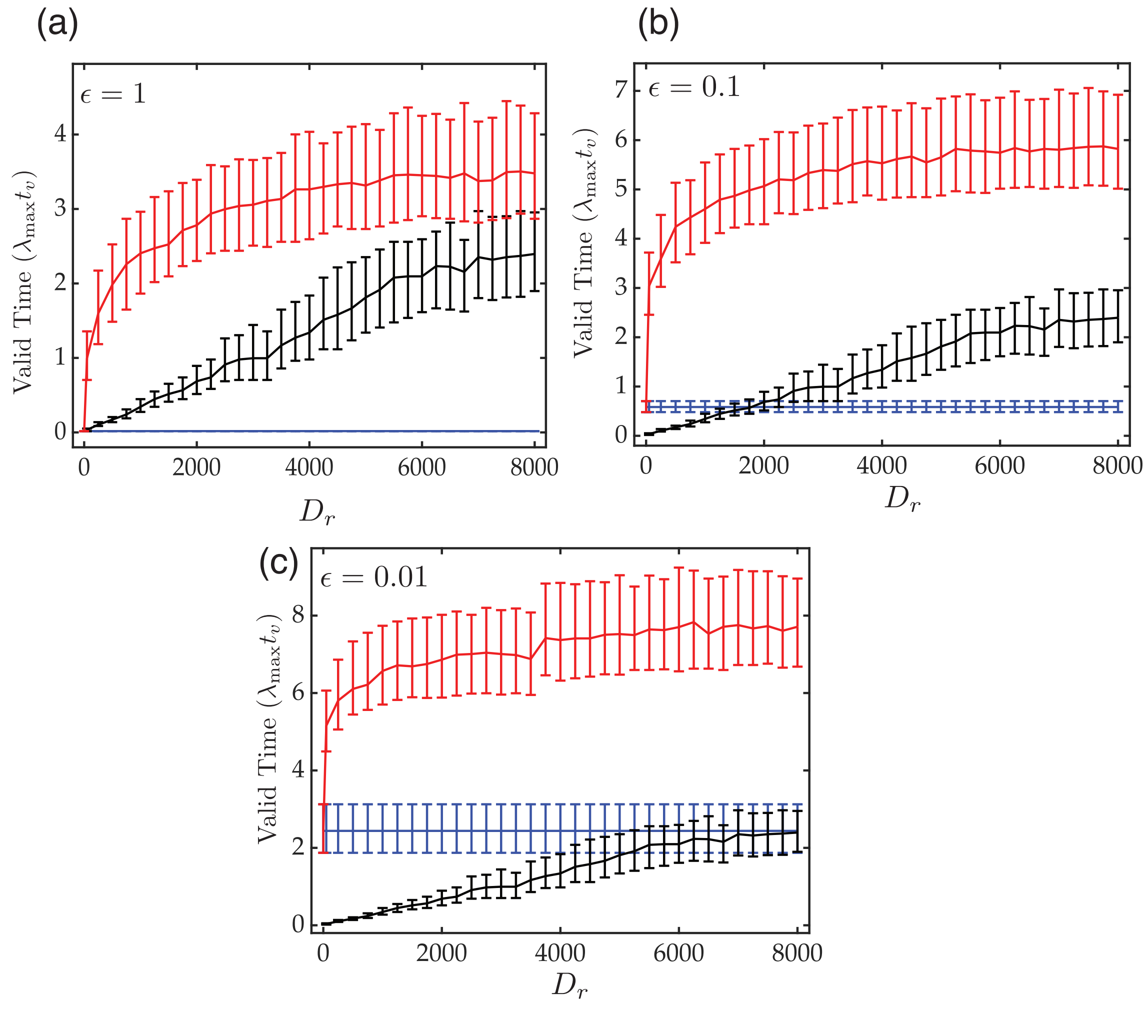


Figure 4.8: Each of the three panels (a), (b), and (c) shows a comparison of the KS system prediction performance of the reservoir-only scheme (black), the knowledge-based model (blue) and the hybrid scheme (red). The median valid time in Lyapunov units ($\lambda_{max} t_v$) is plotted against the size of the reservoir used in the hybrid scheme and the reservoir-only scheme. Since the knowledge-based model does not use a reservoir, its valid time does not vary with the reservoir size. The error in the knowledge-based model is $\epsilon = 1$ in panel (a), $\epsilon = 0.1$ in panel (b) and $\epsilon = 0.01$ in panel (c).

Parameter	Value	Parameter	Value
ρ	0.4	T	5000
$\langle d \rangle$	3	γ	0.5
σ	1.0	τ	250
Δt	0.25	ξ	10

Table 4.2: Reservoir parameters ρ, $\langle d \rangle$, σ, Δt, training time T, hybrid parameter γ, and evaluation parameters τ, ξ for the KS system prediction.

This latter remarkable result is reinforced by Fig. 4.8(a), which shows that even for very large error, $\epsilon = 1$, such that the model is totally ineffective, the hybrid of these two methods is able to predict for a significant amount of time using a relatively small reservoir. This implies that a non-viable model can be made viable via the addition of a reservoir component of modest size. Further Figs. 4.8(b,c) show that even if one has a model that can outperform the reservoir prediction, as is the case for $\epsilon = 0.01$ for most reservoir sizes, one can still benefit from a reservoir using our hybrid technique.

4.6 Conclusions

In this chapter we present a method for the prediction of chaotic dynamical systems that hybridizes reservoir computing and knowledge-based prediction. Our main results are:

1. Our hybrid technique consistently outperforms its component reservoir-only or knowledge-based model prediction methods in the duration of its ability to accurately predict, for both the Lorenz system and the spatiotemporal chaotic Kuramoto-Sivashinsky equations.

2. Our hybrid technique robustly yields improved performance even when the reservoir-only predictor and the knowledge-based model are so flawed that they do not make accurate predictions on their own.

3. Even when the knowledge-based model used in the hybrid is significantly flawed, the hybrid technique can, at small reservoir sizes, make predictions

comparable to those made by much larger reservoir-only predictors, which can be used to save computational resources.

4. Both the hybrid scheme and the reservoir-only predictor have the property of "training reusability" (Sec. 4.3.2), meaning that once trained, they can make any number of subsequent predictions (without retraining each time) by preceding each such prediction with a short run in the training configuration (see Figs. 4.1 and 4.2) in order to resynchronize the reservoir dynamics with the dynamics to be predicted.

Chapter 5: Reservoir observers: Model-free inference of unmeasured variables in chaotic system

This chapter is based on work in the paper, 'Reservoir observers: Model-free inference of unmeasured variables in chaotic systems', Chaos 27, 041102 (2017); https://doi.org/10.1063/1.4979665, by Zhixin Lu, Jaideep Pathak, Brian Hunt, Michelle Girvan, Roger Brockett, and Edward Ott, ©by the American Institute of Physics

Knowing the state of a dynamical system as it evolves in time is important for a variety of applications. This chapter proposes a general-purpose method for inferring unmeasured state variables from a limited set of ongoing measurements. Our method is intended for situations in which mathematical models of system dynamics are unavailable or are insufficiently accurate to perform the desired inference. We use the machine-learning technique called "reservoir computing," with which we construct a system-independent means of processing the measurements. A key point is the extent to which this approach is "universal." That is, our examples show that the same reservoir can be trained to infer the state of different systems. It is the training that relates to a specific system, not the "hardware." The reservoir hardware plays a similar role to an animal's brain, which retrains itself as the system represented by its body and environment changes.

5.1 Introduction

Frequently, when studying the dynamics of a physical system, one only has access to a limited set of measurements of the state variables and desires to deduce unmeasured state variables. In principle, it might be possible to accomplish this goal if, in addition to the measurements, one also has knowledge of the system dynamics. In control theory, a successful deduction method of this type this is called an *observer*. Observers are of great utility for control and prediction of dynamics. The observer problem for the case in which the dynamical system is linear was fully solved in the classic work of Kalman, who also formulated conditions for "observability" under which it is possible to achieve the goal of deducing the full state of a linear system from a given partial set of state measurements (see textbooks on control theory, e.g., Ref. [69]). Observers and observability have also been extensively investigated for nonlinear dynamical systems (e.g., Ref. [70]). For example, in situations of chaotic dynamics, an approach using synchronization of chaos has been exploited [71, 72].

In this chapter we consider the observer problem for situations in which one does not have a sufficient accurate mathematical model of the nonlinear system of interest. In place of such a model, we assume that there exists an initial period of time for which measurements of all the desired system variables are available, and we seek to use these initial measurements in the initial period of time to deduce the full set of desired variables for the subsequent time, for which we assume that measurements of only a limited subset of the desired variables are possible. Our method utilizes a machine learning technique, called *reservoir computing* (see Ref. [1]). This technique uses an input/output neural network with randomly generated parameters, and uses linear regression to choose “output weights” that fit the raw network output to a set of “training data”. We use the data from the initial period of full measurement as the training data. Then we continue to input the subsequent partial set of continually measured variables, and regard the weighted network output as the estimated values of the variables that are no longer measured.

The main result of this chapter is that this kind of “reservoir observer,” subject to certain limitations, can be extremely effective. In what follows we first describe a specific illustrative implementation and review relevant reservoir computing concepts. We then discuss three examples of chaotic systems that highlight the strength and limitations of our method: (1) the Rössler system [73], for which we have done an intensive study of how results depend on design parameters of the reservoir observer; (2) the Lorenz system [66], which we use to illustrate an instance of the issue of observability for our method; and (3) the Kuramoto-Sivashinsky partial differential equation [67, 68, 74], which we use to illustrate the possible effectiveness of our method in cases of spatiotemporal chaos.

5.2 Setup

We consider a dynamical system $d\boldsymbol{\phi}/dt = \mathbf{f}(\boldsymbol{\phi})$ together with a pair of $\boldsymbol{\phi}$-dependent, vector valued variables, $\mathbf{u} = \mathbf{h_1}(\boldsymbol{\phi}) \in \mathbb{R}^M$ and $\boldsymbol{s} = \mathbf{h_2}(\boldsymbol{\phi}) \in \mathbb{R}^P$. We are interested in the situation in which $\mathbf{u}$ and $\mathbf{s}$ can both be measured over a specific period, $[0, T]$, but that only $\mathbf{u}$ can be measured from that time forward; we seek a method for using the continued knowledge of $\mathbf{u}$ to determine $\mathbf{s}$ as a function of time when direct measurement of $\mathbf{s}$ is not available, $t > T$. In contrast with most of the engineering literature devoted to problems of this kind, we do not assume knowledge of $\mathbf{f}$ but rather seek to infer the necessary information from the trajectories recorded on the interval $[0, T]$.

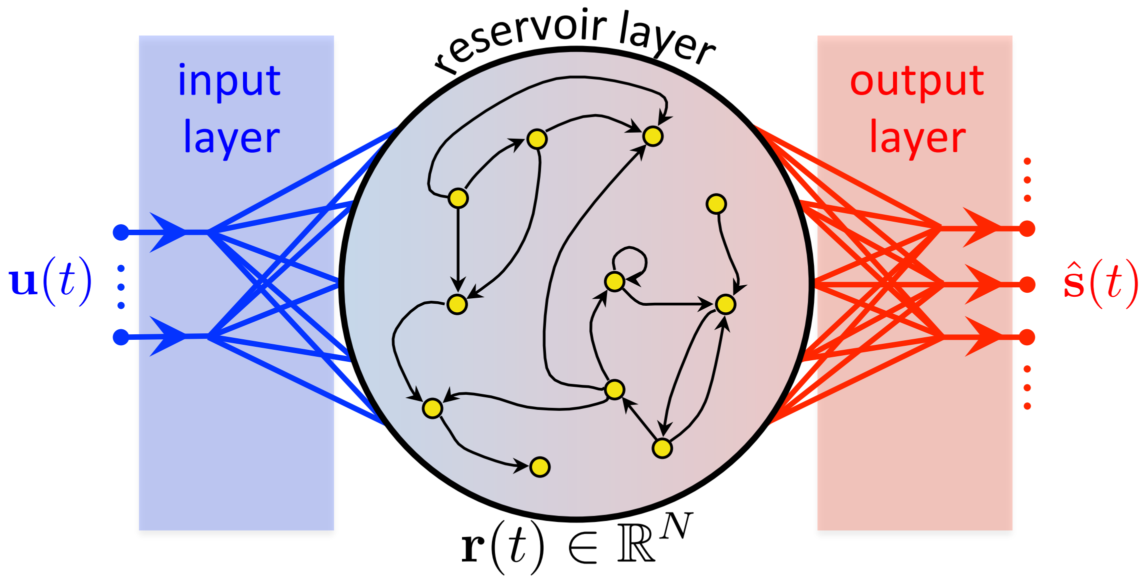


Figure 5.1: A reservoir computer consisting of three parts, an input layer, a reservoir layer with state $\mathbf{r}(t)$, and an output layer. For $t > T$, the input to the system is $\mathbf{u}(t)$ and our goal is that the output $\hat{\mathbf{s}}(t)$ is a good approximation to the unmeasured quantity $\mathbf{s}(t)$.

For this purpose we use "reservoir computing" [1], which has previously been advocated for application to many tasks (e.g., prediction of time series, pattern recognition, etc. []). There are many variations in implementation; in this chapter we adopt the reservoir technique proposed by Jaeger [11]. The reservoir computer has three components (Fig. 5.1), a linear input layer with M input nodes (one for each component of $\mathbf{u}$), a recurrent, nonlinear reservoir network with N dynamical reservoir nodes whose state vector is $\mathbf{r} \in \mathbb{R}^N$, and a linear output layer with P output nodes, as shown in Fig. 5.1. For the specific reservoir computing implementation we use in this chapter, the reservoir dynamics is defined as

$$\mathbf{r}(t+\Delta t) = (1-\alpha)\mathbf{r}(t) + \alpha \tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t) + \xi\mathbf{1}), \tag{5.1}$$

where $\mathbf{A}$ is the (typically sparse) weighted adjacency matrix of the reservoir layer, and the M-dimensional input $\mathbf{u}(t)$ is fed in to the N reservoir nodes via a linear input weight matrix denoted by $\mathbf{W}_{in} \in \mathbb{R}^{N\times M}$ (Note that, in Sec. 5.3.1, $M \geq 1$ and the input $\mathbf{u}$ is a vector). The parameter $0 < \alpha \leq 1$ is a "leakage" rate [75] that controls the time-scale of the reservoir dynamics. We also use a bias term $\xi\mathbf{1}$, where $\mathbf{1}$ is the N-by-1 vector of ones and ξ is a scalar constant. The notation $\tanh(\cdot)$ with a vector argument is defined as the vector whose components are the hyperbolic tangents of the corresponding components of the argument vector. The output, which is a P-dimensional vector, is taken to be a linear function of the reservoir state,

$$\hat{\mathbf{s}}(t) = \mathbf{W}_{out}\mathbf{r}(t) + \mathbf{c}. \tag{5.2}$$

As compared to other artificial neural network approaches, the advantage of reservoir computing is that training is made computationally feasible for relatively large N, since only the output weights $\mathbf{W}_{out}$ and the vector $\mathbf{c}$ are adjusted by the training process. (The input weight matrix $\mathbf{W}_{in}$ and the reservoir adjacency matrix $\mathbf{A}$ are initially randomly drawn and then fixed.) A key point is that the reservoir layer serves as an active medium driven by inputs $\mathbf{u}(t)$ where each reservoir node has a different nonlinear response to its inputs, so that for $N \gg 1$ we can hope that almost a wide variety of desired outputs can be approximated by a linear combination of the N-dimensional reservoir nodal response states.

In addition to the parameters Δt, α, and ξ in Eq. (5.1), and the reservoir size N, the reservoir dynamics depend on the parameters p, ρ, and σ, which govern the random generation of $\mathbf{A}$ and $\mathbf{W_{in}}$ as follows. The adjacency matrix $\mathbf{A}$ is built from a sparse random Erdős-Rényi matrix in which the fraction of nonzero matrix elements is p. The values of non-zero elements are randomly drawn independently from a uniform distribution between -1 and 1. We then uniformly rescale all the elements of $\mathbf{A}$ (i.e., multiply $\mathbf{A}$ by a positive scalar) so that the largest value of the magnitudes of its eigenvalues becomes ρ, which we refer to as the "spectral radius" of $\mathbf{A}$. For the input layer, the ith of the M input signals is connected to N/M reservoir nodes with connection weights in the ith column of $\mathbf{W}_{in}$. Each reservoir node receives input from exactly one input signal. The non-zero elements of $\mathbf{W}_{in}$ are randomly chosen from a uniform distribution in $[-\sigma, \sigma]$.

For the convenience of comparing the reservoir performances, we preprocess all the components of $\mathbf{u}(t)$ and $\mathbf{s}(t)$ so that they have zero mean and unit variance.

Starting from a random initial state $\mathbf{r}(-\tau)$, the reservoir evolves following Eq. (5.1) with input $\mathbf{u}(t)$. Here τ is a transient time, chosen large enough to make the reservoir state essentially independent of its initial state by time $t = 0$. We then record the $K = T/\Delta t$ reservoir states for $0 < t \leq T$,

$$\{\mathbf{r}(\Delta t), \mathbf{r}(2\Delta t), ..., \mathbf{r}(T)\}, \tag{5.3}$$

and the concurrent measurements of the state variables that are unmeasured for $t > T$,

$$\{\mathbf{s}(\Delta t), \mathbf{s}(2\Delta t), ..., \mathbf{s}(T)\}. \tag{5.4}$$

We then train the network by choosing the output layer quantities $\mathbf{W_{out}}$ and $\mathbf{c}$ by choosing them so that the reservoir output approximates the measurement for $0 < t \leq T$. We do this by minimizing the following quadratic form with respect to $\mathbf{W_{out}}$ and $\mathbf{c}$,

$$\{\sum_{k=1}^{K} \|\mathbf{W}_{out}\mathbf{r}(k\Delta t) + \mathbf{c} - \mathbf{s}(k\Delta t)\|^2\} + \beta[\mathrm{Tr}(\mathbf{W}_{out}\mathbf{W}_{out}^T)], \tag{5.5}$$

where $\|\mathbf{q}\|^2 = \mathbf{q}^T\mathbf{q}$ for $\mathbf{q}$ a vector. The second term of Eq. (5.5), $\beta[\mathrm{Tr}(\mathbf{W}_{out}\mathbf{W}_{out}^T)]$, is a regularization term included to avoid overfitting $\mathbf{W}_{out}$, where $\beta > 0$ (typically a small number) is the "ridge regression parameter".

If the training is successful, the readout of the reservoir output should yield a good approximation (denoted $\hat{\mathbf{s}}(t)$) to the desired unmeasured quantity $\mathbf{s}(t)$ for

$t > T$. Referring to Eq. (5.2),

$$\hat{\mathbf{s}}(t) = \mathbf{W}^*_{out}\mathbf{r}(t) + \mathbf{c}^*, \tag{5.6}$$

where $\mathbf{W}^*_{out}$ and $\mathbf{c}^*$ denote the solutions for the minimizers of Eq. (5.5),

$$\mathbf{W}^*_{out} = \delta\mathbf{S}\delta\mathbf{R}^T(\delta\mathbf{R}\delta\mathbf{R}^T + \beta\mathbf{I})^{-1}, \tag{5.7}$$

$$\mathbf{c}^* = -[\mathbf{W}^*_{out}\bar{\mathbf{r}} - \bar{\mathbf{s}}], \tag{5.8}$$

$$\bar{\mathbf{r}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{r}(k\Delta t),\ \ \bar{\mathbf{s}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{s}(k\Delta t) \tag{5.9}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix, $\delta\mathbf{R}$ (respectively, $\delta\mathbf{S}$) is the matrix whose kth column is $\mathbf{r}(k\Delta t) - \bar{\mathbf{r}}$ (respectively, $\mathbf{s}(k\Delta t) - \bar{\mathbf{r}}$).

We remark that Eq. (5.1) represents a special choice for the form of the reservoir that is convenient for our purposes. More generally Eq. (5.1) can be expressed as

$$\mathbf{r}(t + \Delta t) = \mathbf{g}(\mathbf{r}(t), \mathbf{u}(t)), \tag{5.10}$$

and other forms of $\mathbf{g}$, different from the choice Eq. (5.1), have been employed for reservoir methods designed to implement goals different from the observer goal that we address here. For example, experimental reservoir implementations have been reported where the dynamics Eq. (5.10) were from an optical network of semiconductor lasers [76], a delay system with a single nonlinear node [77], a field-programmable gate array (FPGA) [78], phase-delay electro-optic devices [79], and even a bucket of water [80], among others. Such choices might also work for a reservoir-based observer and may offer advantages such as the potential for huge increase in speed [79]. The main requirement on the observer dynamics Eq. (5.10) seems to be that it is sufficiently complex and that the dimension of the reservoir state vector $\mathbf{r}$ is sufficiently large that the output Eq. (5.2) can be made to approximate the desired time series (for our goal, $\mathbf{s}(t)$) by adjustment of $\mathbf{W}_{out}$ and $\mathbf{c}$.

5.3 Examples

5.3.1 Kuramoto-Sivashinsky Equations

We now investigate the possibility of using the reservoir-based observation method of Sec. 5.2 to infer estimates of the state of a spatiotemporally chaotic system from spatially sparse measurements without the use of a model.

For this purpose, we test our model-free observation method on simulated data from the Kuramoto-Sivashinsky equation [81] for the scalar variable $y(x,t)$,

$$y_t = -yy_x - y_{xx} - y_{xxxx}. \tag{5.11}$$

We impose spatially periodic boundary conditions, i.e., $y(x+L,t) = y(x,t)$ on a domain $\{x \in (0,L)\}$ of size $L = 22$ and integrate Eq. (5.11) from a randomly chosen initial condition. The integration was performed on an evenly spaced grid of size $Q = 65$. The simulated data consists of Q time series with a time step $\Delta t = 0.25$ units. We denote this set of time series at the Q grid points by the vector $\mathbf{y}(t) = (y_1(t), y_2(t), \cdots, y_Q(t))^T$ with $y_i(t) = y(i\Delta x), \Delta x = L/(Q-1)$. Let $\mathbf{u}(t) = (u_1(t), u_2(t), \cdots u_M(t))^T$ be a vector containing M out of the Q time series in $\mathbf{y}(t)$. In terms of the variables $\mathbf{s}(t)$ and $\mathbf{u}(t)$ introduced in Sec. 5.2, the vector $\mathbf{u}(t)$ represents spatially sparse measurements performed at evenly spaced points on the grid. We denote the rest of the time series by $\mathbf{s}(t)$. We will vary the number of measurements M in the interval $1 \leq M \leq 8$. We assume that we have access to the full set of state measurements $\mathbf{y}(t)$ for the 'training period', $0 \leq t \leq T$. We further assume that after the training period, i.e., for $t > T$, the observer can only access the partial set of system state variables $\mathbf{u}(t)$. The goal is to infer the set of variables $\mathbf{s}(t)$ from the partial measurements $\mathbf{u}(t)$ for $t > T$.

The reservoir observer setup is described in Sec. 5.2 with the identification $Q = M + P$. For the results in this subsection, we use the following set of reservoir parameters,

$$\begin{array}{rcll}
\text{number of reservoir nodes:} & N & = & 3000, \\
\text{spectral radius:} & \rho & = & 0.9, \\
\text{fraction of non-zero links:} & p & = & 0.02, \\
\text{scale of input weights:} & \sigma & = & 0.5, \\
\text{bias constant:} & \xi & = & 0.0, \\
\text{leakage rate:} & \alpha & = & 0.3, \\
\text{time interval:} & \Delta t & = & 0.25, \\
\text{length of training phase:} & T & = & 15000.
\end{array} \tag{5.12}$$

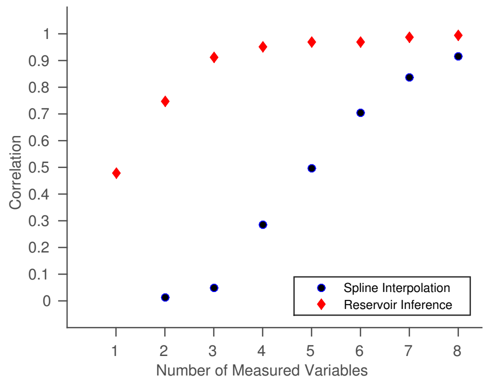


Figure 5.2: Correlation between observer inferred data and the actual data for the Kuramoto-Sivashinsky state versus the number of measured variables M.

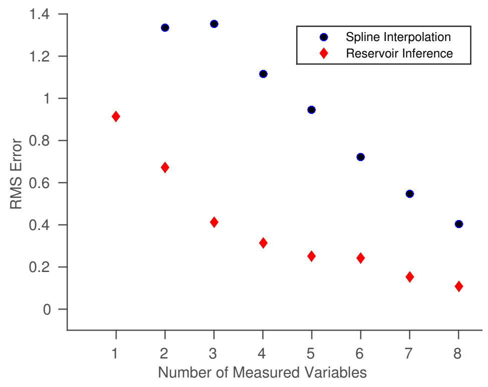


Figure 5.3: RMS error in the inference of the Kuramoto-Sivashinsky state variables versus the number of measured variables M.

To test the quality of the inference for $t \geq T$, we compare the inferred data from the reservoir $\hat{\mathbf{s}}(t)$ with the data series $\mathbf{s}(t)$ obtained from integrating Eq. (5.11). We calculate the correlation between the real and inferred data, defined by

$$C = \frac{\left(\sum_{i,t}(s_i(t) - \langle s\rangle)(\hat{s}_i(t) - \langle \hat{s}\rangle)\right)}{\sqrt{\left(\sum_{i,t}(s_i(t) - \langle s\rangle)^2\right)\left(\sum_{i,t}(\hat{s}_i(t) - \langle \hat{s}\rangle)^2\right)}} \tag{5.13}$$

and the spatially averaged RMS error, defined by

$$R = \sqrt{\frac{\sum_{i,t}\left[s_i(t) - \hat{s}_i(t)\right]^2}{\sum_{i,t}[s_i(t)]^2}}. \tag{5.14}$$

Since the performance of the reservoir may depend on the particular random instance of the reservoir network that is used, we report the mean RMS error and the correlation between the inferred and actual data obtained from 20 observer trials each performing the inference task on the the same data using an independent random realization of the reservoir observer setup. As a baseline comparison, we also report the RMS error and correlation coefficient values for an inference of the unmeasured variables obtained by cubic spline interpolation from the measured variables. Figure 5.2 shows the correlation between the reservoir inference and the actual data as we vary the number of measured variables. The correlation coefficient for the reservoir observer inference is compared with the corresponding value for the cubic spline interpolation scheme. Figure 5.3 shows the RMS error obtained by the reservoir setup as we vary the number of measured variables. Figure 5.4 show comparisons between the actual data and the reservoir inference for $M = 2$ and $M = 4$. These figures demonstrate that, as expected, spline interpolation yields good results at high measurement densities, but that, at lower measurement densities, where spline interpolation yields poor results, the reservoir observer can continue to function well.

5.4 Conclusions

In this chapter we investigate the application of reservoir computing to infer unmeasured state variables of a chaotic dynamical system from a limited set of continually measured state variables for situations in which a mathematical model of the dynamical system is unavailable or is insufficiently accurate. Our main result

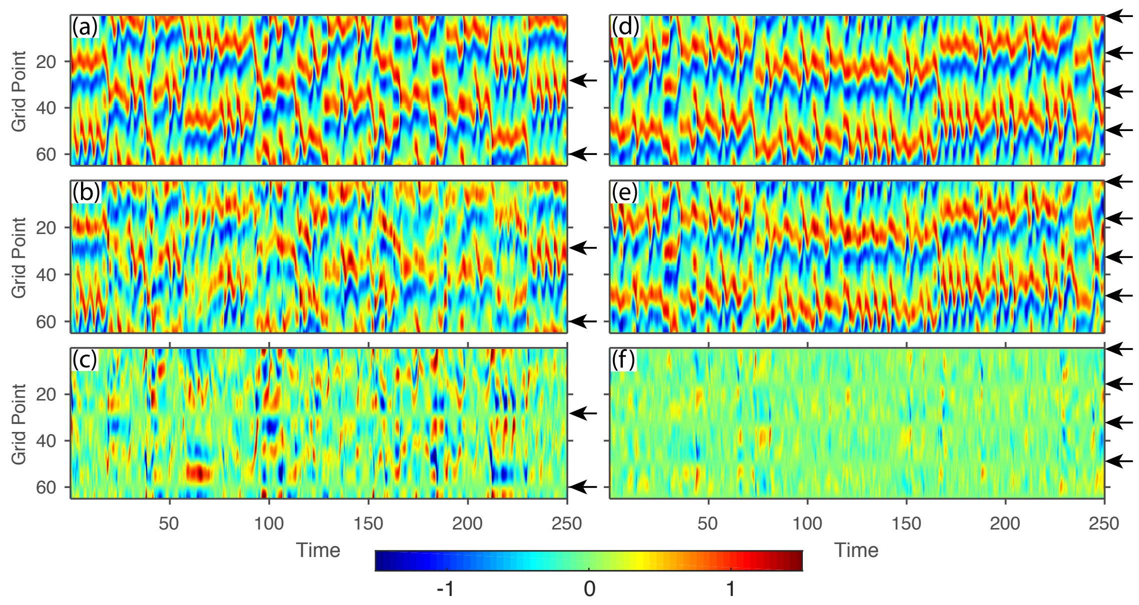


Figure 5.4: Results from two simulations, (a,b,c) and (d,e,f), where (a,b,c) have $M = 2$ inputs, and (d,e,f) have $M = 4$ inputs, whose locations are indicated by the black arrows. The top panels, (a) and (d), show the actual state evolution $y(x, t)$ of the Kuramoto-Sivashinsky system. The middle panels, (b) and (e), show the evolution of the state inferred by the reservoir observer from the measurements. The bottom panels, (c) and (f), show the difference between the inferred data and the actual data.

is that the use of reservoir computers for inference of spatiotemporally chaotic states from spatially sparse data was proposed and validated by application to an example (Sec. 5.3.1).

5.5 Acknowledgment

We are thankful to Daniel Gauthier and Alex Hartemink for their useful comments and discussions. This work was supported by the Army Research Office under grant W911NF-12-1-0101.

Chapter 6: Reconstruction and Forecasting of Dynamical Systems using Partial Measurements, Imperfect Modeling and Machine Learning Assisted Data Assimilation

6.1 Introduction

This chapter describes a technique for improving forecasts of a chaotic dynamical system when only partial measurements of the system are available. We consider simple toy dynamical models of chaotic systems and make a number of simplifying assumptions and intend this chapter as an exploratory foray into the combination of Data Assimilation and Machine Learning. While some of the simplifying assumptions may not hold true in real-life scenarios such as weather prediction, we believe that this study will pave the way for a better understanding of machine learning architectures that will support weather forecasting applications. We emphasize that a number of advances will be required before such a technique can be operationally used. We hope that this chapter and the promising results contained therein provides the impetus for research into this highly challenging task.

6.2 Method

We consider a dynamical system represented by the equation,

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}[\mathbf{x}(t)]. \tag{6.1}$$

In Eq. (6.1), $\mathbf{x}(t)$ represents the full state of the dynamical system. We assume that we are only able to obtain partial measurements of the full state $\mathbf{x}$ at regular time intervals. We denote the measurements (or observations) by $\mathbf{y}(k)$, so that

$$\mathbf{y}(k) = \mathbf{H}\mathbf{x}(k\Delta t) + \eta_k. \tag{6.2}$$

In Eq. (6.2), $\mathbf{H}$ denotes the measurement operator. In the experiments performed in this chapter we will assume that $\mathbf{H}$ takes the form of a projection operator, linearly projecting out a subset of the set of full state variables $\mathbf{x}$. A nonlinear measurement operator could be allowed, if supported by the Data Assimilation method. The variable η_k represents normally distributed random noise with mean $\mathbf{0}$ and covariance matrix $\mathbf{R}$. We assume that T past observations are available and denote them by $\{\mathbf{y}_k\}_{-T\leq k<0}$.

In a practical scenario, Eq. (6.1) would represent a real-world dynamical system (such as the Earth's atmosphere), and $\mathbf{y}$ would represent partial noisy measurements of the full state obtained at regular time intervals. However, most often we do not have full knowledge of the dynamical system and as such, Eq. (6.1) is unknown.

What is available is an imperfect model of the dynamical equations which contains unavoidable model error. This model error is typically due to lack of knowledge of all the physical processes that govern the dynamical system. We assume that we have access to such an imperfect knowledge-based model denoted by $\mathbf{G}$ so that the dynamical equation

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{G}[\mathbf{x}(t)] \tag{6.3}$$

can be used to forecast the future state of the dynamical system $\mathbf{x}$. In practice, the initial state of the dynamical system, $\mathbf{x}$, is estimated from current observation $\mathbf{y}_0$ and all past observations $\{\mathbf{y}_k\}_{-T\leq k<0}$.

6.2.1 Data Assimilation

Data assimilation is a technique that seeks to construct the 'best' possible estimate of a dynamical system based on past observations, a dynamical evolution model and the current observation. Given a dynamical model (Eq. 6.3) and a set of observations $\{\mathbf{y}_k\}_{-T\leq k\leq 0}$, the goal of data assimilation is to find a trajectory $\mathbf{x}(\mathbf{t})$ of the model $\mathbf{G}$ that minimizes the cost function given by

$$J(\mathbf{x}(t)) = \sum_{j=-T}^{0} \left[\mathbf{y}_j - \mathbf{H}\mathbf{x}(j\Delta t)\right] \mathbf{R}^{-1} \left[\mathbf{y}_j - \mathbf{H}\mathbf{x}(j\Delta t)\right]. \tag{6.4}$$

We need to algorithmically compute or approximate the value of $\mathbf{x}(0)$ corresponding to the trajectory that minimizes $\mathbf{J}$. In the situation where the measurement/observation errors (η_k) are Gaussian, the model $\mathbf{G}$ is a perfect representation of the dynamical system and the dynamical system is linear, minimizing Eq. 6.4 will give us the trajectory of the model $\mathbf{x}(t)$ that is the most likely trajectory of the dynamical system in the Bayesian sense.

6.2.2 Kalman Filter: Linear Case

We now consider the case where the model $\mathbf{G}$ is linear and a perfect representation of the dynamical system and the measurement noise η_k has a Gaussian distribution $\eta \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. A Kalman Filter recursively determines the best fit trajectory described in Sec. 6.2.1. One starts with an initial guess for the state $\mathbf{x}_{-T}$. The guess is refined using the observation $\mathbf{y}_{-T}$. At each subsequent time j in $-T \leq j \leq 0$, we use the current best guess $\mathbf{x}_j$ and refine it using the observation $\mathbf{y}_j$. The process can be mathematically described as follows.

Let $\mathbf{x}^a_{j-1}$ be the best guess of the state of the dynamical system at time $j-1$. Thus, $\mathbf{x}^a_{j-1}$ is the mean of a Gaussian probability distribution that represents the likelihood of the states given the observations upto (and including) time $j-1$. Let $\mathbf{P}^a_{j-1}$ be the covariance of this probability distribution. Following the assumption of linearity, a Gaussian distribution $\left(\mathbf{x}^a_{j-1}, \mathbf{P}^a_{j-1}\right)$ propagates to another Gaussian distribution $\left(\mathbf{x}^b_j, \mathbf{P}^b_j\right)$ when the model acts on it, where,

$$\mathbf{x}^b_j = \mathbf{G}\mathbf{x}^a_{j-1} \tag{6.5}$$

$$\mathbf{P}^b_j = \mathbf{G}\mathbf{P}^a_{j-1}\mathbf{G}^T. \tag{6.6}$$

It can be shown that the distribution that best represents the state of the dynamical system at time j is then given by a Gaussian distribution $\left(\mathbf{x}_j^a, \mathbf{P}_j^a\right)$, where

$$\mathbf{P}_j^a = (I + \mathbf{P}_j^b \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{P}_j^b \tag{6.7}$$

$$\mathbf{x}_j^a = \mathbf{x}_j^b + \mathbf{P}_j^a \mathbf{H}^T \mathbf{R}^{-1} (\mathbf{y}_j - \mathbf{H}\mathbf{x}_j^b) \tag{6.8}$$

6.2.3 Kalman Filter: Nonlinear Case

If the model $\mathbf{G}$ is nonlinear, then the assumptions made in arriving at Eqs. (6.7,6.8) do not hold. Particularly, one can no longer propagate the distributions between times $j-1$ and j using Eqs. (6.5, 6.6). Several approaches have been studied for extending the Kalman Filter to the case where the dynamical system is nonlinear. One approach known as the Extended Kalman Filter linearizes the model $\mathbf{G}$ and uses the linearized model in the state propagation equations (Eqs. 6.5, 6.6). The calculations involved in the linearization of $\mathbf{G}$ as well as in the computation and propagation of the covariance matrix from such a linearization are usually not feasible unless the model is low-dimensional. The Ensemble Kalman Filter (EnKF) [82] was proposed To overcome this computational difficulty. Several improvements have been made to the EnKF since it was first proposed. Particularly, the Ensemble Transform Kalman Filter (ETKF) [83,84] is a major computational improvement over the EnKF. The ETKF can be computationally parallelized using the Local Ensemble Transform Kalman Filter (LETKF) algrithm of Refs [85,86] which evolved from the Local Ensemble Kalman Filter algorithm [87]. We will first discuss the core idea of the EnKF and then go on to describe the ETKF algorithm. In this chapter, we will be use the ETKF algorithm as the baseline for determining forecast quality. In Sec. 6.3, we will modify the ETKF using Machine Learning.

In the EnKF, one keeps track of an ensemble of states $\{\mathbf{x}_j^{a,i}\}_{1\leq i\leq E}$ that represent a sampling from the probability distribution of the 'best' state at time j. The ensemble is propagated using the model dynamics so that

$$\mathbf{x}_j^{b,i} = \hat{\mathbf{G}}_{j-1}\mathbf{x}_{j-1}^{a,i}. \tag{6.9}$$

Here, $\hat{\mathbf{G}}_j$ represents the nonlinear operator that propagates a state vector at time j to the state vector at time $j+1$ according to the model dynamics. Further, the covariance of the probability distribution corresponding to an ensemble can be estimated as,

$$\mathbf{P}_j^b = (E-1)^{-1}\sum_{i=1}^{E}(\mathbf{x}_j^{b,i} - \bar{\mathbf{x}}_j^b)(\mathbf{x}_j^{b,i} - \bar{\mathbf{x}}_j^b)^T \tag{6.10}$$

Where, $\bar{\mathbf{x}}_j^b$ represents the mean of the ensemble members $\{\mathbf{x}_j^{b,i}\}_{1\leq i\leq E}$. Thus, the covariance matrix is evolved implicitly. The ensemble $\{\mathbf{x}_j^{a,i}\}$ (which represents a sample from the probability distribution of the best guess for the state at time j) is called the analysis ensemble at time j. The ensemble $\{\mathbf{x}_j^{b,i}\}$ (which represents a sample from the probability distribution of the forecast at time j from the analysis ensemble at the previous time step $j-1$) is called the background ensemble at time j. Similarly $\mathbf{P}_j^a$ ($\mathbf{P}_j^b$) is called the analysis (background) covariance. As in Sec. 6.2.2

that our goal is to find the analysis ensemble and the from the background ensemble. In the earliest version of the EnKF, the analysis mean and covariance were obtained from the background mean and covariance using Eqs. (6.7, 6.8). However, it was shown in Ref. [88] that, for theoretical consistency, an ensemble of observations needs to be created by perturbing the observations with random noise. The Kalman updates then need to be applied to each individual ensemble member using a perturbed observation.

We now describe the steps involved in the ETKF analysis step without theoretical analysis or justification. These steps follow Ref. [85] and the reader is encouraged to follow the theoretical derivation of the ETKF in that reference.

1. Create the matrix

$$\mathbf{X}_j^b = \left[\mathbf{x}_j^{b,1} - \bar{\mathbf{x}}_j^b | \mathbf{x}_j^{b,2} - \bar{\mathbf{x}}_j^b | \ldots | \mathbf{x}_j^{b,E} - \bar{\mathbf{x}}_j^b\right], \tag{6.11}$$

   whose columns represent deviations of the ensemble members from the ensemble mean.

2. Compute the matrix $\mathbf{Y}_j^b = \mathbf{H}\mathbf{X}_j^b$

3. Create a matrix

$$\tilde{\mathbf{P}}_j^a = \left[(E-1)I/\rho + \mathbf{C}\mathbf{Y}_j^b\right] \tag{6.12}$$

   where $\mathbf{C} = (\mathbf{Y}_j^b)^T\mathbf{R}^{-1}$ and ρ is a parameter called the 'covariance inflation'.

This parameter is a multiplicative factor greater than unity that expands the covariance of the analysis ensemble in an ad-hoc manner. It is essential since the theoretical analysis which guarantees that the analysis ensemble covariance represents the true covariance of the model state are true only in the case of the linear Kalman filter. The covariance inflation has to be tuned to optimize the accuracy of the ETKF according to some forecast quality metric.

4. Compute $\mathbf{W}_j^a = \left[(E-1)\tilde{\mathbf{P}}_j^a\right]^{1/2}$, where the $[.]^{1/2}$ denotes the symmetric square root.

5. Compute $\bar{\mathbf{w}}_j^a = \tilde{\mathbf{P}}^a\mathbf{C}(\mathbf{y}_j - \mathbf{y}_j^b)$ and add it to each column of $\mathbf{W}_j^a$. Each column of $\mathbf{W}_j^a$ is now denoted by $\mathbf{w}_j^{a,i}$

6. Compute $\mathbf{x}_j^{a,i} = \mathbf{X}_j^b\mathbf{w}_j^{a,i} + \mathbf{x}_j^b$. The analysis ensemble is given by $\{\mathbf{x}_j^{a,i}\}_{1\leq i\leq E}$

The steps above represent the analysis step, i.e., obtaining the analysis ensemble from the background ensemble. The analysis ensemble obtained at the end of the above steps is then evolved using the model dynamics to obtain the background ensemble at the next time step after which the analysis step is repeated using the observation at that time step. The analysis ensemble is initialized in a fairly arbitrary manner by perturbing any state on the dynamical attractor of the model with gaussian random noise.

6.3 Machine Learning Assisted Ensemble Transform Kalman Filtering

In Sec. 6.2.2 and Sec. 6.2.3 we assume that the model **G** represents the actual dynamics of the system (Eq. 6.1) fairly accurately. In other words, we assume that the model error is small. If, however, we are unable to construct a good enough model, then the forecast quality will suffer. In Chapter 4, we described a machine learning technique for correcting imperfections in a model. However we assumed that we were able to observe to the full state of the dynamical system i.e., the measurement operator **H** was assumed to be an identity matrix. In this section, we relax that assumption and let **H** be a projection operator so that our measurements are a subset of the system variables. We now describe our algorithm for our Machine Learning Assisted Ensemble Transform Kalman Filter. In Sec. 6.4 we will compare the performance of the ML-ETKF with the traditional ETKF algorithm and compare forecasts made by the model from the optimized analysis state in both cases.

6.3.1 Reservoir Computer

The reservoir computer implementation is similar to Chapter 4. The reservoir network adjacency matrix, denoted **A**, is a $D_r \times D_r$ sized sparse randomly generated matrix. The network is constructed to have an average degree denoted by $\langle d \rangle$, and the nonzero elements of **A**, representing the edge weights in the network, are initially chosen independently from the uniform distribution over the interval $[0, 1]$. All the edge weights in the network are then uniformly scaled via multiplication of the adjacency matrix by a constant factor to set the largest magnitude eigenvalue of the matrix to a quantity ω, which is called the 'spectral radius' of $\mathbf{A}$. The state of the reservoir, given by the vector $\mathbf{r}(t)$, consists of the components r_j for $1 \leq j \leq D_r$ where $r_j(t)$ denotes the scalar state of the j^{th} node in the network. The reservoir is coupled to the M dimensional input through a $D_r \times M$ dimensional matrix $\mathbf{W}_{in}$. Each row of the matrix $\mathbf{W}_{in}$ has exactly one randomly chosen nonzero element. Each nonzero element of the matrix is independently chosen from the uniform distribution on the interval $[-\zeta, \zeta]$.

6.3.2 Algorithm

As outlined in Sec. 6.2, we assume that the true dynamical system is given by Eq. (6.1). We have T measurements given by $\{\mathbf{y}_j\}$, in the interval $-T \leq j \leq 0$ which are related to the true state of the dynamical system by Eq. 6.2. We further assume that the model $\mathbf{G}$ (Eq. 6.3) is imperfect. Using the model $\mathbf{G}$ and the ETKF algorithm outlined in Sec. 6.2.3, we can obtain an analysis state $\mathbf{x}_j^a$ at each time step j, $-T \leq j \leq 0$. We thus create a set of analysis states $\{\mathbf{x}_j^a\}_{-T \leq j \leq 0}$. We are interested in forecasting the state of the dynamical system for $j > 0$. In the standard ETKF setup, one would predict the future state of the dynamical system (Eq. 6.1) using $\mathbf{x}_0^a$ as the initial condition and Eq. 6.3 as the dynamical model. We will call this forecast the ETKF forecast. The ETKF forecast is the baseline against which we will evaluate the forecast made by our ML-ETKF algorithm which we will now describe.

6.3.2.1 Training

1. Use the model $\mathbf{G}$ to create a set of forecasts from each of the analysis states $\mathbf{x}_j^a$, $-T \leq j \leq 0$. We will denote these forecasts by $\tilde{\mathbf{x}}_j$, $-T+1 \leq j \leq 1$. Table 6.1 provides an easy way to visualize the states and the forecasts indexed by time j.

2. Initialize the reservoir state to a random value. We will index this reservoir state to the time index $j = -T$ and denote this initial reservoir state $\mathbf{r}_{-T}$.

3. Evolve the reservoir computer using the following equation

$$\mathbf{r}_{j+1} = \tanh[\mathbf{A}\mathbf{r}_j + \mathbf{W}_{in}\tilde{\mathbf{x}}_{j+1}] \tag{6.13}$$

   for $-T \leq j \leq 0$.

4. Find a set of output weights $\mathbf{W}_{out}$ so that

$$\mathbf{W}_{out} \begin{bmatrix} \mathbf{r}_j \\ \tilde{\mathbf{x}}_j \end{bmatrix} \simeq \mathbf{y}_j \tag{6.14}$$

   for $-T+1 \leq j \leq 0$. The matrix $\mathbf{W}_{out}$ is computed using regularized least squares regression. Thus, we find the $\mathbf{W}_{out}$ that minimizes the following L_2

j	$-T$	$-T+1$	$-T+2$	$-T+3$	$\cdots$	-2	-1	0	1
analysis	$\mathbf{x}^a_{-T}$	$\mathbf{x}^a_{-T+1}$	$\mathbf{x}^a_{-T+2}$	$\mathbf{x}^a_{-T+3}$	$\cdots$	$\mathbf{x}^a_{-2}$	$\mathbf{x}^a_{-1}$	$\mathbf{x}^a_0$	
forecast		$\tilde{\mathbf{x}}_{-T+1}$	$\tilde{\mathbf{x}}_{-T+2}$	$\tilde{\mathbf{x}}_{-T+3}$	$\cdots$	$\tilde{\mathbf{x}}_{-2}$	$\tilde{\mathbf{x}}_{-1}$	$\tilde{\mathbf{x}}_0$	$\tilde{\mathbf{x}}_1$
reservoir	$\mathbf{r}_{-T}$	$\mathbf{r}_{-T+1}$	$\mathbf{r}_{-T+2}$	$\mathbf{r}_{-T+3}$	$\cdots$	$\mathbf{r}_{-2}$	$\mathbf{r}_{-1}$	$\mathbf{r}_0$	$\mathbf{r}_1$
observations	$\mathbf{y}_{-T}$	$\mathbf{y}_{-T+1}$	$\mathbf{y}_{-T+2}$	$\mathbf{y}_{-T+3}$	$\cdots$	$\mathbf{y}_{-2}$	$\mathbf{y}_{-1}$	$\mathbf{y}_0$	

Table 6.1: Chronologically indexed analysis states of the imperfect model ($\mathbf{x}^a_j$), imperfect model forecasts ($\tilde{\mathbf{x}}_j$), reservoir states ($\mathbf{r}_j$) and observations ($\mathbf{y}_j$). The time index $j = 0$ represents the present time. The time interval $-T \leq j \leq 0$ represents the past during which observations (and thus, analysis states) are available. All time steps $j > 0$ are considered to be in the future. No observations are available from the future and thus, cannot be part of our training data set.

norm

$$\ell(\mathbf{W}_{out}) = \sum_{j=-T+1}^{0} \|\mathbf{W}_{out} \begin{bmatrix} \mathbf{r}_j \\ \tilde{\mathbf{x}}_j \end{bmatrix} - \mathbf{y}_j\|^2 + \beta\|\mathbf{W}_{out}\|^2 \tag{6.15}$$

6.3.2.2 Prediction

We now describe the ML-ETKF prediction algorithm that uses the trained reservoir along with ETKF to forecast the state of the dynamical system from time $j = 0$ onward.

Initial Prediction

1. compute the ML forecast of the observation at time $j = 1$. Note that since the current time is $j = 0$, the actual observation at $t = 1$ is unavailable. The

ML forecast of the observation, $\tilde{\mathbf{y}}_1$ is given by

$$\mathbf{y}_1^{ML} = \mathbf{W}_{out} \begin{bmatrix} \mathbf{r}_1 \\ \\ \tilde{\mathbf{x}}_1 \end{bmatrix} \tag{6.16}$$

2. Propagate the analysis ensemble $\{\mathbf{x}_0^{a,i}\}$, to obtain the background ensemble $\{\mathbf{x}_1^{b,i}\}$ at time $j = 1$ by using the model dynamics $\mathbf{G}$.

3. Perform steps 1 to 4 of the ETKF algorithm.

4. Perform step 5 of the ETKF algorithm with $\mathbf{y}_1^{ML}$ instead of $\mathbf{y}_1$ ($\mathbf{y}_1$ is unavailable at time $j = 0$).

5. Perform step 6 of the ETKF algorithm to obtain the analysis ensemble at time $j = 1$. Since this step is not truly an analysis step we call it pseudo-assimilation. This is due to the fact that a real observation was not used. Instead we relied on an ML predicted observation. We denote this analysis ensemble by $\{\mathbf{s}_1^{a,i}\}_{1\le i\le E}$. The mean of this ensemble $\mathbf{s}_1^a$ is our ML-ETKF forecast of the dynamical system at time $j = 1$.

Subsequent Predictions At the end of step 5 above, we have the pseudo-analysis ensemble $\{\mathbf{s}_1^{a,i}\}$ and the pseudo-analysis mean $\mathbf{s}_1^a$. At time $j \geq 2$ we assume that we have the pseudo-analysis ensemble $\{\mathbf{s}_{j-1}^{a,i}\}$ and the pseudo-analysis mean $\mathbf{s}_{j-1}^a$

1. Use the model $\mathbf{G}$ to propagate the pseudo-analysis mean $\mathbf{s}_{j-1}^a$ to $\tilde{\mathbf{s}}_j$.

2. Perform the reservoir update

$$\mathbf{r}_j = \tanh[\mathbf{A}r_{j-1} + \mathbf{W}_{in}\tilde{\mathbf{s}}_j] \tag{6.17}$$

3. compute the ML forecast of the observation at time j, $\mathbf{y}_j^{ML}$, given by

$$\mathbf{y}_j^{ML} = \mathbf{W}_{out} \begin{bmatrix} \mathbf{r}_j \\ \\ \tilde{\mathbf{s}}_j \end{bmatrix} \tag{6.18}$$

4. Propagate the ensemble $\{\mathbf{s}_{j-1}^{a,i}\}$ to the ensemble $\{\mathbf{s}_j^{b,i}\}$ using the model dynamics $\mathbf{G}$.

5. Perform steps 1 to 4 of the ETKF algorithm on the ensemble $\{\mathbf{s}_{j-1}^{b,i}\}$.

6. Perform step 5 of the ETKF algorithm substituting $\mathbf{y}_j^{ML}$ instead of $\mathbf{y}_j$ (The real observation $\mathbf{y}_j$ is unavailable at time $j = 0$).

7. Perform step 6 of the ETKF algorithm to obtain the pseudo-analysis ensemble at time j. Thus, we have obtained $\{\mathbf{s}_j^{a,i}\}$. The pseudo-analysis mean $\bar{\mathbf{s}}_j^a$ is the ML-ETKF forecast at time j. At the end of this step we have $\{\mathbf{s}_j^{b,i}\}$. We also obtained $\tilde{\mathbf{s}}_j$ in step 1. We can now increment j by one and repeat steps 1 to 7.

6.4 Results

We demonstrate the performance of the ML-ETKF algorithm in comparison with the baseline ETKF algorithm for forecasting the state of a dynamical system. We consider two dynamical systems as our examples, the Lorenz ‘63 system [66] and the Kuramoto-Sivashinsky (KS) system [67, 68]. Simulated data is generated from the true model dynamics in the time interval $-T \leq j \leq P$. The data in the interval $1 \leq j \leq P$ is set aside to test the quality of forecasts made by the ETKF and ML-ETKF schemes but is not otherwise used since this data is considered to be part of the future. We further assume that we only have access to the observations $\{\mathbf{y}_j\}_{-T\leq j\leq 0}$ and do not know the perfect model equations (Eq. 6.1). We assume that we have access to an imperfect model $\mathbf{G}$ (Eq. 6.3). We will use this imperfect model to test the ETKF and ML-ETKF schemes and compare forecasts made by the two schemes for $j > 0$.

6.4.0.1 Baseline ETKF Forecast:

Using the ETKF scheme as described in Sec. 6.2.3, we use the observations $\mathbf{y}_j$ $(-T \leq j \leq 0)$ and the model $\mathbf{G}$ to arrive at the analysis state $\mathbf{x}_0^a$. Using $\mathbf{x}_0^a$ as the initial condition, obtain a forecast using the model $\mathbf{G}$ from time $j = 1$ to $j = P$. We call this forecast $\mathbf{x}_j^{f,base}$, $1 \leq j \leq P$.

Hyperparameter	Lorenz 63	KS
D_r	1000	2000
$\langle d \rangle$	3	3
ω	0.9	0.6
ζ	0.1	1.0

Table 6.2: Reservoir Hyperparamenters

6.4.0.2 ML-ETKF Forecast:

We follow the ML-ETKF scheme detailed in Sec. 6.3 using a reservoir computer with the parameters listed in Table 6.2 for the Lorenz '63 and the Kuramoto Sivashinsky dynamical systems. We use the observations $\mathbf{y}_j$ $(-T \leq j \leq 0)$ and the imperfect model $\mathbf{G}$ corresponding to the Lorenz and KS systems respectively to train the reservoir computer. The exact form of the observations and of the imperfect model is described in Sections 6.4.1, 6.4.2 We then forecast using the ML-ETKF scheme from $j = 1$ to $j = P$. We call this forecast $\mathbf{x}_j^{f,ml}$, $1 \leq j \leq P$.

6.4.0.3 RMS error

The RMS error in the baseline ETKF forecast $(\mathbf{e}_j^{base})$ and the RMS error in ML-ETKF forecast $(\mathbf{e}_j^{ml})$ are calculated as follows:

$$\mathbf{e}_j^{base(ml)} = \frac{\|\mathbf{x}_j^{f,base(ml)} - \mathbf{x}_j\|}{\langle \|\mathbf{x}_j\| \rangle} \qquad 1 \leq j \leq P \tag{6.19}$$

6.4.0.4 Valid Time

We define the Valid Time (VT) as the time j at which the RMS error $(\mathbf{e}_j^{base(ml)})$ exceeds a threshold κ chosen to be 0.9.

6.4.1 Lorenz 63

The Lorenz system is described by the equations,

$$\frac{dX_1}{dt} = -aX_1 + aX_2 \tag{6.20}$$

$$\frac{dX_2}{dt} = bX_1 - X_2 - X_1X_3 \tag{6.21}$$

$$\frac{dX_3}{dt} = -cX_3 + X_1X_2 \tag{6.22}$$

where $a = 10$, $b = 8/3$, and $c = 28$. In this example, we let the true dynamical system of Eq. 6.1 be the system given by Eqs. (6.20). We obtain simulated data by integrating Eqs. (6.20) using a fourth order Runge-Kutta solver. We sample the time series at intervals $\Delta t = 0.1$. This data represents the truth $\mathbf{x}(j\Delta t)$, $-T \leq j \leq P$. We use the data in the interval $-T \leq j \leq 1$ to create simulated observations

$$\mathbf{y}_j = \mathbf{H}^1\mathbf{x}_j + \eta_j. \tag{6.23}$$

Here $\mathbf{H}^1$ represents the operator that projects out the variable X_1 from $[X_1; X_2; X_3]$ so that

$$\mathbf{H}^1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{6.24}$$

The imperfect model, $\mathbf{G}$, is assumed to differ from the true dynamics in a single parameter value. We let the imperfect model equations be given by,

$$\frac{dX_1}{dt} = -aX_1 + aX_2, \tag{6.25}$$

$$\frac{dX_2}{dt} = b(1+\epsilon)X_1 - X_2 - X_1X_3, \tag{6.26}$$

$$\frac{dX_3}{dt} = -cX_3 + X_1X_2. \tag{6.27}$$

Thus, the imperfect model $\mathbf{G}$ differs from the perfect model $\mathbf{F}$ in the parameter b by a multiplicative factor $(1+\epsilon)$.

6.4.1.1 Results: Optimizing the Covariance Inflation

The forecast quality of both the ETKF and ML-ETKF algorithms is strongly dependent on the covariance inflation parameter ρ. It is thus crucial that the covariance inflation factor is optimized independently for both the ETKF and ML-ETKF forecast schemes. In Fig. 6.1, we demonstrate the dependence of the forecast Valid Time on the Ensemble Covariance Inflation factor ρ. We perform 20 independent predictions each using a dataset generated from a different initial condition. We do this at different values of the covariance inflation parameter ρ. We demonstrate this dependence on ρ when the Model Error $\epsilon = 0.1$. Figure 6.1 shows the valid time for a set of forecasts made with the baseline ETKF scheme (red markers) and the ML-ETKF scheme (blue markers). We also plot the median valid time for both of the schemes and denote the median valid time with a larger marker. Thus, Fig. 6.1 shows that the ML-ETKF scheme dramatically improves forecast valid time when the model has substantial error.

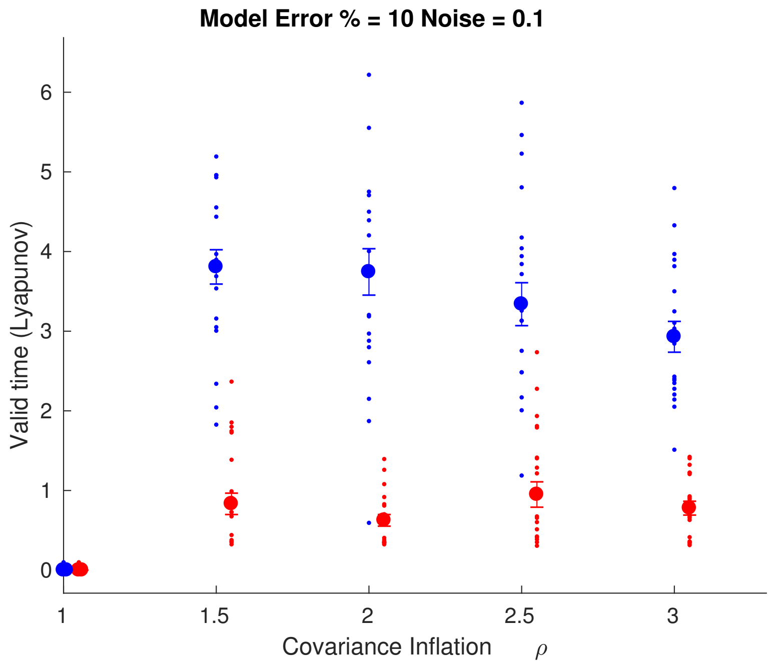


Figure 6.1: Dependence of the Forecast Valid Time for the Lorenz 63 model on the covariance inflation factor (ρ). We find that the forecast valid time depends on choosing the correct covariance inflation parameter and it is thus essential to tune the parameter correctly. We see that the ML-ETKF (blue markers) outperforms the baseline ETKF forecast significantly and has a much higher forecast valid time.

6.4.2 Kuramoto-Sivashinsky (KS) system

In this section we consider an example of a spatiotemporal chaotic dynamical system called the the Kuramoto-Sivashinsky (KS) system defined by the Partial Differential Equation (PDE),

$$\frac{\partial u(x,t)}{\partial t} = u\frac{\partial u(x,t)}{\partial x} + \frac{\partial^2 u(x,t)}{\partial x^2} + \frac{\partial^4 u(x,t)}{\partial x^4}. \tag{6.28}$$

Equation (6.28) defines the evolution of a one-dimensional spatiotemporal scalar field $u(x,t)$ defined in the spatial domain $x \in [0, L)$. We assume periodic boundary conditions so that $u(x+L,t) = u(x,t)$. In this example we let Eq. (6.28) represent the true dynamical system $\mathbf{F}$ corresponding to Eq. (6.1). We obtain simulated data by integrating Eq. (6.28) using a pseudo-spectral PDE solver [33]. The domain $[0, L)$ is discretized into Q grid-points. We sample the time series at intervals of $\Delta t = 0.25$. Thus, our simulated data takes the form of a Q-dimensional vector $\mathbf{x}(j\Delta t)$, $-T \leq j \leq P$. As in the previous example of the Lorenz '63 dynamical system (Sec. 6.4.1), we use the data in the interval $-T \leq j \leq 0$ as the training data for the ML-ETKF scheme as well as for and set aside the data in the interval $1 \leq j \leq P$ for forecast verification.

We assume that we have access to an imperfect model, $\mathbf{G}$, of the KS dynamical system given by the equations,

$$\frac{\partial u(x,t)}{\partial t} = u\frac{\partial u(x,t)}{\partial x} + (1+\epsilon)\frac{\partial^2 u(x,t)}{\partial x^2} + \frac{\partial^4 u(x,t)}{\partial x^4}. \tag{6.29}$$

Thus, if $\epsilon = 0$, then the model represents the true dynamics perfectly and a nonzero value of ϵ indicates an imperfect model. We also assume that our measurements are of the form given by Eq. (6.2) The measurement operator $\mathbf{H}$ is a projection operator that projects out Θ uniformly spaced variables (of the Q total variables) in $\mathbf{x}_j$.

6.4.2.1 Results: Dependence on Model Error

To demonstrate the effectiveness of the ML-ETKF technique, we consider a KS system with $L = 35$, $Q = 64$ and a measurement operator $\mathbf{H}$ that measures at $\Theta = 16$ uniformly spaced grid points. Figure 6.2 shows the prediction valid time for 20 independent predictions made with the ML-ETKF and ETKF schemes at four different values of the Model Error ϵ. We see that the ML-ETKF scheme is far superior to the traditional ETKF scheme when the error in the imperfect model is high. On the other hand, as expected, the ETKF algorithm performs about as well as the ML-ETKF algorithm when the Model Error is small. This result is in line with our expectations since the ML-ETKF algorithm is trained to improve the forecast valid time by correcting for model inaccuracies.

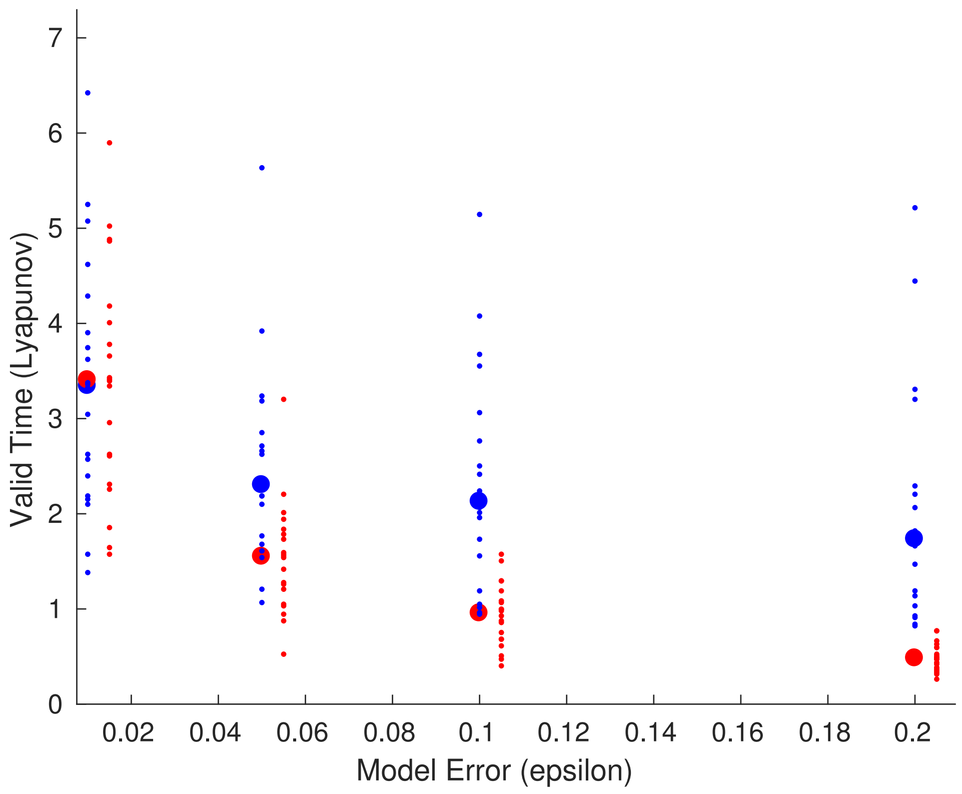


Figure 6.2: Forecast Valid Time of the ML-ETKF (blue markers) and ETKF (red markers) schemes at different values of the Model Error (ϵ). The ML-ETKF scheme vastly outperforms the baseline ETKF scheme at larger values of the model error.

Bibliography

[1] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

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

[3] Tien-Yien Li and James A Yorke. Period three implies chaos. *The American Mathematical Monthly*, 82(10):985–992, 1975.

[4] Robert M May. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560):459–467, 1976.

[5] Stephen Smale. Differentiable dynamical systems. *Bulletin of the American mathematical Society*, 73(6):747–817, 1967.

[6] Edward Ott and Mark Spano. Controlling chaos. In *AIP Conference Proceedings*, volume 375, pages 92–103. AIP, 1996.

[7] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[8] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[9] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[13] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

[14] Ulrich Parlitz and Christian Merkwirth. Prediction of spatiotemporal time series based on reconstructed local states. *Physical review letters*, 84(9):1890, 2000.

[15] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[16] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[19] Sarah Marzen. Difference between memory and prediction in linear recurrent networks. *Physical Review E*, 96(3):032308, 2017.

[20] Masanobu Inubushi and Kazuyuki Yoshimura. Reservoir computing beyond memory-nonlinearity trade-off. *Scientific Reports*, 7(1):10199, 2017.

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

[22] Zhixin Lu, Jaideep Pathak, Brian Hunt, Michelle Girvan, Roger Brockett, and Edward Ott. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(4):041102, 2017.

[23] Xin Yan and Xiaogang Su. *Linear regression analysis: theory and computing.* World Scientific, 2009.

[24] Kristof Vandoorne, Joni Dambre, David Verstraeten, Benjamin Schrauwen, and Peter Bienstman. Parallel reservoir computing using optical amplifiers. *IEEE transactions on neural networks*, 22(9):1469–1481, 2011.

[25] Daniel Brunner, Miguel C Soriano, Claudio R Mirasso, and Ingo Fischer. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nature communications*, 4:1364, 2013.

[26] Laurent Larger, Antonio Baylón-Fuentes, Romain Martinenghi, Vladimir S. Udaltsov, Yanne K. Chembo, and Maxime Jacquot. High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification. *Phys. Rev. X*, 7:011015, Feb 2017.

[27] Lennert Appeltant, Miguel Cornelles Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, Joni Dambre, Benjamin Schrauwen, Claudio R Mirasso, and Ingo Fischer. Information processing using a single dynamical node as complex system. *Nature communications*, 2:468, 2011.

[28] Nicholas D Haynes, Miguel C Soriano, David P Rosin, Ingo Fischer, and Daniel J Gauthier. Reservoir computing with a single time-delay autonomous boolean node. *Physical Review E*, 91(2):020801, 2015.

[29] Paul Manneville. Liapounov exponents for the kuramoto-sivashinsky model. *Macroscopic Modelling of Turbulent Flows*, pages 319–326, 1985.

[30] James L Kaplan and James A Yorke. Chaotic behavior of multidimensional difference equations. In *Functional Differential equations and approximation of fixed points*, pages 204–227. Springer, 1979.

[31] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.*, 120:024102, Jan 2018.

[32] Nikolaevich Tikhonov, Andreĭ, Vasiliĭ Iakovlevich Arsenin, and Fritz John. *Solutions of ill-posed problems*, volume 14. Winston Washington, DC, 1977.

[33] Aly-Khan Kassam and Lloyd N Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

[34] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge University Press, 2004.

[35] Edward Ott, Tim Sauer, and James A Yorke. Coping with chaos. analysis of chaotic data and the exploitation of chaotic systems. *Wiley Series in Nonlinear Science, New York: John Wiley*, 1994.

[36] Henry Abarbanel. *Analysis of observed chaotic data.* Springer Science & Business Media, 2012.

[37] Floris Takens. Lecture notes in mathematics. *by DA Rand and L.-S. Young Springer, Berlin*, 898:366, 1981.

[38] Tim Sauer, James A Yorke, and Martin Casdagli. Embedology. *Journal of Statistical Physics*, 65(3):579–616, 1991.

[39] Dr S Broomhead and Gregory P King. Extracting qualitative dynamics from experimental data. *Physica D: Nonlinear Phenomena*, 20(2-3):217–236, 1986.

[40] Anke Brandstater and Harry L Swinney. Strange attractors in weakly turbulent couette-taylor flow. *Physical Review A*, 35(5):2207, 1987.

[41] J-P Eckmann, S Oliffson Kamphorst, David Ruelle, and S Ciliberto. Liapunov exponents from time series. *Physical Review A*, 34(6):4971, 1986.

[42] Valery Petrov, Vilmos Gaspar, Jonathan Masere, and Kenneth Showalter. Controlling chaos in the belousovzhabotinsky reaction. *Nature*, 361(6409):240–243, 1993.

[43] Mantas Lukosevivcius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[44] Bruce Ira Cohen, JA Krommes, WM Tang, and MN Rosenbluth. Non-linear saturation of the dissipative trapped-ion mode by mode coupling. *Nuclear Fusion*, 16(6):971, 1976.

[45] Yoshiki Kuramoto and Toshio Tsuzuki. Persistent propagation of concentration waves in dissipative media far from thermal equilibrium. *Progress of Theoretical Physics*, 55(2):356–369, 1976.

[46] GI Sivashinsky. Large cells in nonlinear marangoni convection. *Physica D: Nonlinear Phenomena*, 4(2):227–235, 1982.

[47] Mark C Cross and Pierre C Hohenberg. Pattern formation outside of equilibrium. *Reviews of Modern Physics*, 65(3):851, 1993.

[48] R Livi, A Politi, and S Ruffo. Distribution of characteristic exponents in the thermodynamic limit. *Journal of Physics A: Mathematical and General*, 19(11):2033, 1986.

[49] David A Egolf and Henry S Greenside. Relation between fractal dimension and spatial correlation length for extensive chaos. *Nature*, 369(6476):129–131, 1994.

[50] Arkady Pikovsky and Antonio Politi. Dynamic localization of lyapunov vectors in spacetime chaos. *Nonlinearity*, 11(4):1049, 1998.

[51] Holger Kantz, Gunter Radons, and Hongliu Yang. The problem of spurious lyapunov exponents in time series analysis and its solution by covariant lyapunov vectors. *Journal of Physics A: Mathematical and Theoretical*, 46(25):254009, 2013.

[52] Timothy D Sauer, Joshua A Tempkin, and James A Yorke. Spurious lyapunov exponents in attractor reconstruction. *Physical Review Letters*, 81(20):4341, 1998.

[53] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[54] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[55] Wolfgang Maass, Thomas Natschlager, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[56] Leonard A Smith. Intrinsic limits on dimension calculations. *Physics Letters A*, 133(6):283–288, 1988.

[57] J-P Eckmann and David Ruelle. Fundamental limitations for estimating dimensions and lyapunov exponents in dynamical systems. In *Turbulence, Strange Attractors And Chaos*, pages 447–449. World Scientific, 1995.

[58] Taeshik Shon and Jongsub Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799–3821, 2007.

[59] Chih-Fong Tsai and Ming-Lun Chen. Credit rating by hybrid machine learning techniques. *Applied soft computing*, 10(2):374–380, 2010.

[60] Dimitris C Psichogios and Lyle H Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

[61] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[62] Jaideep Pathak, Zhixin Lu, Brian R Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[63] Laurent Larger, Miguel C Soriano, Daniel Brunner, Lennert Appeltant, Jose M Gutiérrez, Luis Pesquera, Claudio R Mirasso, and Ingo Fischer. Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. *Optics express*, 20(3):3241–3249, 2012.

[64] Laurent Larger, Antonio Baylón-Fuentes, Romain Martinenghi, Vladimir S Udaltsov, Yanne K Chembo, and Maxime Jacquot. High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification. *Physical Review X*, 7(1):011015, 2017.

[65] Piotr Antonik, Marc Haelterman, and Serge Massar. Brain-inspired photonic signal processor for generating periodic patterns and emulating chaotic systems. *Physical Review Applied*, 7(5):054014, 2017.

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

[67] Yoshiki Kuramoto and Toshio Tsuzuki. Persistent propagation of concentration waves in dissipative media far from thermal equilibrium. *Progress of theoretical physics*, 55(2):356–369, 1976.

[68] GI Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flamesi. derivation of basic equations. *Acta astronautica*, 4(11-12):1177–1206, 1977.

[69] Katsuhiko Ogata and Yanjuan Yang. *Modern control engineering*. Prentice-Hall, Upper Saddle River, NJ, 1970.

[70] Robert Hermann and Arthur J Krener. Nonlinear controllability and observability. *IEEE Transactions on automatic control*, 22(5):728–740, 1977.

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

[72] John C Quinn, Paul H Bryant, Daniel R Creveling, Sallee R Klein, and Henry DI Abarbanel. Parameter and state estimation of experimental chaotic systems using synchronization. *Physical Review E*, 80(1):016201, 2009.

[73] Otto E Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[74] Bruce Ira Cohen, JA Krommes, WM Tang, and MN Rosenbluth. Non-linear saturation of the dissipative trapped-ion mode by mode coupling. *Nuclear fusion*, 16(6):971, 1976.

[75] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

[76] Daniel Brunner, Stephan Reitzenstein, and Ingo Fischer. All-optical neuromorphic computing in optical networks of semiconductor lasers. In *Rebooting Computing (ICRC), IEEE International Conference on*, pages 1–2. IEEE, 2016.

[77] Yvan Paquot, Joni Dambre, Benjamin Schrauwen, Marc Haelterman, and Serge Massar. Reservoir computing: a photonic neural network for information processing. In *SPIE Photonics Europe*, pages 77280B–77280B. International Society for Optics and Photonics, 2010.

[78] Nicholas D Haynes, Miguel C Soriano, David P Rosin, Ingo Fischer, and Daniel J Gauthier. Reservoir computing with a single time-delay autonomous boolean node. *Physical Review E*, 91(2):020801, 2015.

[79] Laurent Larger, Antonio Baylón-Fuentes, Romain Martinenghi, Vladimir S. Udaltsov, Yanne K. Chembo, and Maxime Jacquot. High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification. *Phys. Rev. X*, 7:011015, Feb 2017.

[80] Chrisantha Fernando and Sampsa Sojakka. Pattern recognition in a bucket. In *European Conference on Artificial Life*, pages 588–597. Springer, 2003.

[81] James M Hyman and Basil Nicolaenko. The kuramoto-sivashinsky equation: a bridge between pde's and dynamical systems. *Physica D: Nonlinear Phenomena*, 18(1-3):113–126, 1986.

[82] Geir Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[83] Craig H Bishop, Brian J Etherton, and Sharanya J Majumdar. Adaptive sampling with the ensemble transform kalman filter. part i: Theoretical aspects. *Monthly weather review*, 129(3):420–436, 2001.

[84] Xuguang Wang, Craig H Bishop, and Simon J Julier. Which is better, an ensemble of positive–negative pairs or a centered spherical simplex ensemble? *Monthly Weather Review*, 132(7):1590–1605, 2004.

[85] Brian R Hunt, Eric J Kostelich, and Istvan Szunyogh. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter. *Physica D: Nonlinear Phenomena*, 230(1-2):112–126, 2007.

[86] Istvan Szunyogh, Eric J Kostelich, Gyorgyi Gyarmati, Eugenia Kalnay, Brian R Hunt, Edward Ott, Elizabeth Satterfield, and James A Yorke. A local ensemble transform kalman filter data assimilation system for the ncep global model. *Tellus A: Dynamic Meteorology and Oceanography*, 60(1):113–130, 2008.

[87] Edward Ott, Brian R Hunt, Istvan Szunyogh, Aleksey V Zimin, Eric J Kostelich, Matteo Corazza, Eugenia Kalnay, DJ Patil, and James A Yorke. A local ensemble kalman filter for atmospheric data assimilation. *Tellus A: Dynamic Meteorology and Oceanography*, 56(5):415–428, 2004.

[88] Gerrit Burgers, Peter Jan van Leeuwen, and Geir Evensen. Analysis scheme in the ensemble kalman filter. *Monthly weather review*, 126(6):1719–1724, 1998.