Scientific Machine Learning
for Modeling Complex Dynamical Systems and Efficient Data Assimilation

By

Chuanqi Chen

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2026

Date of final oral examination: April 10, 2026

The dissertation is approved by the following members of the Final Oral Committee:

Jinlong Wu, Assistant Professor, Mechanical Engineering
Dan Negrut, Professor, Mechanical Engineering
Jacob Notbohm, Associate Professor, Mechanical Engineering
Wenxiao Pan, Associate Professor, Mechanical Engineering
Nan Chen, Associate Professor, Mathematics

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to those who have contributed to the completion of my doctoral thesis.

First and foremost, I am deeply thankful to my advisor, Dr. Jinlong Wu, for his invaluable guidance, unwavering support, and insightful advice throughout my PhD journey. His systematic training and mentorship have played a pivotal role in my development as a rigorous and logical researcher.

I am also grateful to the members of my dissertation committee—Dr. Dan Negrut, Dr. Jacob Notbohm, Dr. Wenxiao Pan, Dr. Nan Chen, and Dr. Jinlong Wu—for their time, patience, and valuable feedback on this work. Their expertise and constructive criticism have significantly enriched the quality of this thesis.

Next, I would like to extend my deepest gratitude to my lab-mates, collaborators, and friends—Xinghao Dong, Huchen Yang, Zhongrui Wang, Yinling Zhang, Jonah Jeffrey Spencer, and Zisheng Ye—for their exceptional collaboration and unwavering support.

Finally, I extend my heartfelt thanks to my parents, Xiaolin Chen and Jianqin Zhao, for their encouragement and sacrifices. Their boundless support has been the cornerstone of my academic journey, and I am profoundly grateful for their presence in my life.

Contents

List of Tables

List of Figures

Abstract

As an interdisciplinary field, scientific machine learning (SciML) integrates machine learning with scientific computing to study complex dynamical systems, which are ubiquitous across diverse domains such as fluid dynamics, materials science, geophysics, and aerospace. Addressing the intrinsically complex properties of these systems—including strong nonlinearity, high dimensionality, and stochasticity—this thesis focuses on developing reliable and efficient SciML frameworks to solve the associated forward and inverse problems. Specifically, it advances core methodologies in spatiotemporal modeling, system identification, data assimilation, and uncertainty quantification.

The thesis is structured around three primary contributions. First, it investigates the continuous modeling of spatiotemporal dynamical systems with short-term state forecasting and long-term statistics matching. Second, it proposes novel algorithms for the online sparse identification of dynamical systems with regime switching. Third, it designs surrogate frameworks for modeling partially observed nonlinear dynamical systems and efficient nonlinear data assimilation. Across all three parts, the efficacy and efficiency of the proposed methods and algorithms are validated through numerical experiments on canonical complex dynamical systems in science and engineering.

Data assimilation (DA), a principled framework that integrates computational models with observational data for state estimation, establishes a synergistic connection with SciML in this thesis: it acts both as an essential tool for enhancing SciML and as a key application of it. Specifically, DA facilitates the robust identification and learning of SciML models, while SciML provides a highly efficient and scalable computational framework for implementing DA. The joint investigation of SciML and DA promotes the development of both fields and advances the modeling, simulation, and analysis of complex dynamical systems.

In summary, this thesis integrates diverse sources of information—including observational data, physical laws, and domain knowledge—to develop systematic computational

methods on the basis of SciML and DA, encompassing model architectures, learning strategies, and diagnostic processes. These methods contribute to addressing previously intractable and emerging challenges in complex dynamical systems across science and engineering, demonstrating strong potential for real-world applications.

Chapter 1

Introduction

This thesis focuses on the development and application of computational methods and algorithms through a joint investigation of scientific machine learning and data assimilation for the modeling, simulation, and analysis of complex dynamical systems. Such systems are ubiquitous in science and engineering, including fluid dynamics, geophysical and climate systems, biological processes, and engineered systems. Accurate description of those systems and a deep understanding of the underlying mechanisms are essential for enabling reliable prediction, decision-making, and effective control and design across a wide range of applications. However, their high dimensionality, strong nonlinearity, intrinsic stochasticity, and partial observability pose significant challenges. This thesis aims to develop efficient and accurate methodologies for addressing those challenges and solving associated forward and inverse problems, including state forecasting, state estimation, data assimilation, system identification, and uncertainty quantification.

1.1 Complex Dynamical Systems

Complex dynamical systems are ubiquitous across diverse fields such as fluid dynamics, materials science, geophysics, and aerospace, where they are used to describe and characterize physical, biological, and engineering systems [1–6]. Those systems are of fundamental importance because they provide a unified framework for understanding, predicting, and controlling the evolution of real-world systems over time. Accurate modeling of such systems and a deep understanding of their underlying mechanisms enable reliable prediction

of critical phenomena such as turbulent flows and climate variability, thereby supporting control and design, state estimation, anomaly detection, and decision-making in downstream applications. As a result, advancing methodologies for modeling and simulating complex dynamical systems is essential for addressing many pressing challenges in science and engineering.

Despite their importance, modeling complex dynamical systems remains highly challenging due to their intrinsic properties. These systems are often characterized by strong nonlinearity, high dimensionality, multiscale interactions, chaos and turbulence, discontinuities, stochasticity, non-Gaussian statistics, intermittency, and extreme events [7–12]. A conventional approach to describing such systems is to derive governing equations from first principles, typically in the form of nonlinear ordinary and partial differential equations (ODEs and PDEs) and their stochastic extensions (SDEs). While these physics-based models offer key advantages, including adherence to physical laws and strong interpretability, their derivation often requires deep domain knowledge, extensive prior information, and simplifying assumptions. Consequently, models constructed under idealized conditions may fail to capture essential dynamics in real-world systems due to limited expressivity, unknown or unresolved mechanisms, lack of information, and uncertainty. Additionally, computationally expensive numerical simulations are typically required to understand and predict these systems, often rendering practical applications prohibitive.

With the rapid growth of available data and computing resources, data-driven methods have emerged as dominant and cost-effective approaches for modeling and simulating complex dynamical systems. Those methods includes closure models [13, 14], sparse identification [15, 16], reduced order models [17–19], causality-based models [20–22], and Gaussian processes [23]. Although these approaches have demonstrated increasing potential and promise in the study of complex dynamical systems, developing efficient and accurate simulation methods and discovering dynamical models from observational data remain open challenges. In particular, several fundamental issues remain unresolved:

1. **Data-driven spatiotemporal modeling with preserved long-term properties.** There is a lack of modeling frameworks capable of learning complex spatiotemporal dynamical systems (e.g., nonlinear PDEs) from observational data while preserving continuity in both space and time. Such continuity is essential for enabling training and inference across varying spatiotemporal discretizations, thereby achieving

discretization- or resolution-invariance. Furthermore, existing approaches often fail to simultaneously attain accurate short-term state predictions and statistically consistent long-term behavior, which is particularly crucial for chaotic and turbulent systems that are often assumed to be ergodic.

2. **Online identification of regime-switching dynamical systems.** Real-time detection and adaptation to regime switching in dynamical systems—typically characterized by abrupt changes in governing equations or model parameters—remain challenging. Such behavior is ubiquitous in many real-world applications, including robotic locomotion across heterogeneous terrains (e.g., transitions from grass to sand) and atmospheric jet dynamics exhibiting blocked and unblocked regimes. Consequently, there is a need for robust and efficient methodologies that can identify such transitions and update the dynamical model for the new regime from sequential data.

3. **Efficient state estimation and forecasting under partial observations.** For partially observed dynamical systems, particularly those that are strongly nonlinear and high-dimensional, accurately inferring hidden states and generating reliable forecasts from partial observations remains a fundamental challenge. This difficulty arises from incomplete information, the lack of tractable analytical tools, high computational cost, and associated uncertainties. In most real-world systems, the full system state is not directly accessible, and only partial, noisy observations are available. Consequently, there is a critical need for surrogate modeling frameworks tailored to partially observed settings, capable of learning system dynamics from data for simultaneously efficient and accurate state estimation and forecasting.

Addressing these fundamental issues is critical for making methodological contributions to the study of dynamical systems and for solving relevant real-world problems. In practice, observational data are typically heterogeneous including multiple resolutions, non-uniform temporal sampling rates and spatial discretizations, short-term time series, and time-averaged statistics. Consequently, the integration of heterogeneous information sources not only enhances model robustness and reliability, but may also be necessary when individual data sources are insufficient. Furthermore, although the precise long-term evolution of real-world systems such as atmospheric dynamics and turbulent fluid flows is inherently unpredictable, their statistical property and geometric structure often exhibit

consistency and stability. Capturing these invariant properties enables computational models to faithfully represent the underlying dynamics and supports decision-making based on predictions from both short- and long-term behavior. Additionally, the governing dynamics of a system are often not fixed, but evolve due to intrinsic variability or changes in external conditions, leading to regime switching. For example, atmospheric jet streams can meander in different directions as the system alternates between blocked and unblocked regimes. Similarly, excitable media are highly sensitive to finite perturbations, which can trigger transitions from a quiescent state to complex wave patterns. Such regime switching can increase the likelihood of extreme events, giving rise to phenomena such as extreme weather and climate patterns, neuronal bursting, and severe ductile failure in materials. Detecting regime transitions and identifying the associated underlying dynamics, through appropriate model identification methods, therefore has significant scientific and societal importance. Finally, it is usually difficult or even impractical to obtain complete data of system states in real-world applications due to several factors: (i) the system may lack well-defined boundaries or contain hidden states, (ii) certain state variables may be inherently difficult or impossible to measure, and (iii) the digital devices used for data acquisition can have computational or storage constraints. For example, in weather and climate systems, the thermocline depth is typically difficult to observe directly due to its location beneath the ocean surface. As a result, the available data often provide only a partial representation of the underlying system, leading to partially observed dynamical systems. Advanced methods for state estimation and forecasting with uncertainty quantification are essential for addressing the challenges arising from partial observability. This PhD thesis aims to develop systematic and reliable computational frameworks and algorithms to address these challenges in complex dynamical systems.

1.2 Scientific Machine Learning

Scientific machine learning (SciML) is an interdisciplinary field that integrates scientific computing and machine learning with the aim of facilitating the modeling, simulation, and analysis of complex dynamical systems by combining data-driven methods with physical laws and domain knowledge. Apart from the data-driven methods mentioned in Section 1.1, the neural-network-based models from deep learning [24–28] have significantly advanced the field of SciML and emerged as a dominant approach for modeling complex dynamical

systems [29–33]. Based on the universal approximation theorem [34–36], neural networks can approximate any nonlinear functions and operators. Consequently, they exhibit strong expressivity and effectiveness in capturing dynamics and making predictions. For example, as a continuous-in-time version of ResNet [37], the neural ODE [38] is proposed to learn the dynamics of systems with efficient memory usage. Physics-informed neural networks [39], which integrate physical laws into the neural networks, are developed to solve forward and inverse PDE-governed problems. Designed to learn mappings between infinite-dimensional function spaces, neural operators [40–45] have been widely used to approximate solution operators and build continuous spatiotemporal models.

Recently, generative learning [46–50] has been increasingly employed and integrated into the field of SciML. For example, the diffusion models have been utilized to construct stochastic nonlocal models [51], generate spatiotemporal turbulence [52], capture extreme event statistics [53], and learn PDE solutions [54]. Additionally, active learning [55, 56] has gained attention in SciML applications for identifying informative data [57, 58], simulating nonlinear systems [59], estimating model parameters [60], and learning model discrepancies [61]. The aforementioned SciML methods and techniques have demonstrated strong effectiveness in numerical simulation, operator learning, physics-informed data-driven modeling, data acquisition, as well as model discovery, reduction, and calibration for complex dynamical systems and related applications.

However, the problems and challenges outlined at the end of Section 1.1 remain inadequately addressed by current developments in SciML. For instance, data assimilation (DA), a fundamental technique for state estimation in science and engineering, faces significant difficulties when applied to complex dynamical systems characterized by high dimensionality, strong nonlinearity, and non-Gaussian statistics, due to the lack of analytical tools and prohibitive computational costs. To address this issue, a novel SciML surrogate model is proposed to enable efficient and accurate DA for general nonlinear dynamical systems. Conversely, DA serves as a critical tool for advancing SciML models by facilitating the learning of probabilistic relationships between system states and capturing long-term statistical behavior of these systems. In this thesis, SciML and DA form a tightly coupled paradigm, with DA playing an integral role in their synergy for solving forward and inverse problems.

1.3 Data Assimilation

Data assimilation (DA) integrates observational data with computational models to enhance state estimation [62–65]. It is extensively applied in real-time state estimation, parameter estimation, imputation, and optimal control. [66–70]. For complex dynamical systems that are partially observed (e.g., state-space models), the governing equations are utilized as computational models for short-term statistical prediction, known as the prior distribution. Through Bayesian inference, the prior distribution is updated with the observational data to the posterior distribution, which is the result of DA. DA is further classified into three types of problems: filtering, smoothing, and prediction, which correspond to estimating the conditional distributions of the current, past, and future states, respectively, given past and current observations. It is worth noting that prediction via DA fundamentally differs from the classical state forecast problem, in which the initial condition or distribution is assumed to be known. Instead, DA-based prediction is initialized using states estimated through the assimilation process (e.g., filtering). This underscores an important feature of DA (e.g., filtering): its robustness to unknown initial conditions or distributions. Even initialized from a random guess, DA can gradually converge to a stable and accurate estimate by sequentially incorporating observational data, and the initial transient phase of this process is commonly referred to as the warm-up time.

To handle general nonlinear dynamical systems, the ensemble DA methods [71–75], which utilize samples to approximate probabilistic distributions, are commonly used techniques. However, the direct application of ensemble DA methods to high-dimensional dynamical systems is often computationally intractable, primarily due to the substantial computational cost incurred by the curse of dimensionality in sampling processes and the numerical simulations of dynamical models including nonlinear PDEs. Although the adoption of efficient computational models, such as data-driven surrogate models [76–79], reduced-order models [80, 81], and stochastic parameterizations [82–85], coupled with various DA techniques like localization [86, 87] and empirical tunings of ensemble sizes and covariances matrix, can mitigate costs and facilitate the implementation of DA, the DA process still demands significant computational resources, and may suffer from sampling and approximation errors.

Recently, deep learning has been employed to support DA for complex dynamical systems [88], assisting the construction of computational models by neural networks that

enhance forecast capability within DA [89, 90]. Additionally, statistical variational DA methods have been proposed to generate observations in scenarios where observational data is scarce [91]. Using encoder-decoder networks in latent DA has proven effective in utilizing multi-domain data for high-dimensional systems [92]. Deep learning can also facilitate using data-driven ensembles at a low computational cost to accelerate ensemble DA [93]. Furthermore, end-to-end deep learning methods have been developed to streamline the entire DA process [94, 95]. While those methods have facilitated the implementation and application of DA, efficient DA methods for general nonlinear dynamical systems still require further exploration and development. More importantly, this thesis focuses on developing a unified SciML framework capable of both state forecast and efficient DA with a low training cost and robust inference results.

Conversely, DA plays a vital role in the development of SciML methods. Inspired by the ensemble Kalman filter (EnKF) [71, 73, 96], the ensemble Kalman inversion (EKI) [68, 97] serves as a derivative-free optimization method that is widely used for parameter estimation and inverse problems. The derivative-free nature of EKI, which minimizes loss functions without requiring gradients, makes it particularly effective for memory-intensive auto-differentiation and for chaotic systems where gradients may be unstable or unreliable. Additionally, DA can force statistical constraints to guide SciML models toward capturing probabilistic relationships between states and mitigating overfitting [98, 99]. Moreover, effectively leveraging online observational data through DA to enhance the performance of SciML models is essential for their reliable deployment in real-world applications, thereby motivating the integrated study of SciML and DA.

The remainder of the thesis is organized as follows. Chapter 1 introduces the overall background and motivation of the thesis. Chapter 2 proposes a continuous model with a new hybrid optimization scheme for modeling spatiotemporal dynamical systems with short-term state forecasting and long-term statistics matching. Chapter 3 introduces a causality-based method for online sparse identification of dynamical systems with regime switching. Chapter 4 designs a unified deep learning framework for modeling partially observed nonlinear dynamical systems and efficient data assimilation. Chapter 5 summarizes the thesis and highlights its core contributions.

Chapter 2

Data-Driven Spatiotemporal Modeling with Short-Term State Forecasting and Long-Term Statistics Matching

Data-driven modeling techniques have been explored in the spatial-temporal modeling of complex dynamical systems for many engineering applications. However, a systematic approach is still lacking to leverage the information from different types of data, e.g., with different spatial and temporal resolutions, and the combined use of short-term trajectories and long-term statistics. In this work, we build on the recent progress of neural operator and present a data-driven modeling framework called neural dynamical operator that is continuous in both space and time. A key feature of the neural dynamical operator is the resolution-invariance with respect to both spatial and temporal discretizations, without demanding abundant training data in different temporal resolutions. To improve the long-term performance of the calibrated model, we further propose a hybrid optimization scheme that leverages both gradient-based and derivative-free optimization methods and efficiently trains on both short-term time series and long-term statistics. We investigate the performance of the neural dynamical operator with three numerical examples, including the viscous Burgers' equation, the Navier–Stokes equations, and the Kuramoto–Sivashinsky equation. The results confirm the resolution-invariance of the proposed modeling framework and also demonstrate stable long-term simulations with only short-term time series data. In addition, we show that the proposed model can better predict long-term statistics

via the hybrid optimization scheme with a combined use of short-term and long-term data.

2.1 Introduction

Complex dynamical systems are ubiquitous in many science and engineering applications, e.g., mixing phenomena in atmospheric and ocean dynamics [100–102], physical and chemical processes of energy conversion [103], and drones operating in turbulent flows [104]. For most of these systems, numerically simulating the governing equations derived by first principles is still infeasible in the foreseeable future, and thus closure models are inevitably needed to account for those unresolved degrees of freedom. Many of the classical closure models are calibrated with simplified settings and are known to have limited predictive capability, mainly due to the lack of enough expressive power in the model form and the empirical calibration to a small amount of data. In the past few decades, we have witnessed the rapid growth of high-fidelity simulation and experimental data and the great advance of machine learning methods, which motivated the development of data-driven modeling techniques [20, 21, 30, 32, 33, 51, 66, 105–110] with the aim of improving or even replacing the classical models by neural-network-based models.

Machine learning methods have achieved great success in the past decade, e.g., convolutional neural networks [28, 111] for tasks such as computer vision (CV), and recurrent neural networks [112–115] for tasks such as natural language processing (NLP). More recently, transformer-based models have demonstrated even better performance in both CV and NLP tasks. Inspired by the success of these methods in standard machine learning tasks, many previous works explored using these methods to improve the modeling of dynamical systems. Despite the initial promising results, standard machine learning methods often assume fixed discretization in both space and time, which presents a significant constraint for accurately modeling complex dynamical systems. The main reason for such a limitation is two-fold: (i) the available data sources are unlikely in a consistent resolution, and (ii) the efficient characterization of the system state usually demands a non-uniform resolution. For instance, the data sources of earth system include high-fidelity simulations [116] with a fine spatial resolution such as $\mathcal{O}(100\text{m})$ to $\mathcal{O}(1\text{km})$, satellite images across a wide range of resolutions with the finest one of $\mathcal{O}(10\text{m})$, and the data collected by observatories that sparsely distributed across the earth. The temporal resolutions also vary among high-fidelity simulations, satellite images, and observatory data. This multiscale nature of different data

sources is also present in many engineering applications, e.g., wind farms with both LiDAR data and high-fidelity simulations of atmospheric boundary layer [117, 118]. On the other hand, the system states of complex dynamics often have multiscale features intrinsically, for which an adaptive resolution is often the more efficient way of characterizing the system state than a uniform resolution.

The multiscale nature of both the system state and the available data sources of complex dynamical systems has motivated the development of continuous form machine learning methods for the modeling and simulation of those systems. For instance, Fourier neural operator (FNO) [41] employs Fourier and inverse Fourier transformations to construct Fourier layers, which are used as building blocks to approximate an operator (i.e., a mapping between Banach spaces) in the continuous form. Essentially, FNO belongs to a general category of neural operators [119], i.e., using neural networks to approximate the integral operator as the basic component of the approximation of a more general operator. Other neural operator methods that belong to this category are graph neural operators (GNO) [120, 121] and low-rank neural operators (LNO), which connects to another popular neural operator framework known as DeepONet [40, 122]. More specifically, DeepONet introduced branch net and trunk net to implement an approximation of operator based on the universal approximation theorem of operators. There are also other recent works that construct neural-network-based operators via a wavelet-based approximation of integral operators [123] or reconstruction from other types of sub-networks [124]. Comparisons between neural operator approaches have been studied in [119, 125]. The approximation error has been studied in [126] with a residual-based error correction method being proposed. Physics constraints [127, 128] and derivative information [129] were demonstrated as additional information sources that can enhance the performance of neural operator methods. There have been many other interesting extensions of the standard neural operator frameworks, e.g., solving problems on general geometries [130], learning of nonlinear non-autonomous dynamical systems [131], and improving neural operator techniques with the recent success of large language models [132, 133], to name a few. Although these methods can be used to characterize the solution operator of a dynamical system, they only work with a fixed and uniform temporal resolution if standard neural operator methods are used. By taking the prediction lead time as an additional input, the neural operator methods can potentially work with non-uniform temporal resolution but often demand abundant data with different temporal resolutions for the training.

On the other hand, neural ordinary differential equation (ODE) [38] provided a temporally continuous framework of machine learning methods for the modeling and simulation of dynamical systems. Recently, the combined use of neural ODE and neural operator has been explored by [134] in the context of standard machine learning tasks such as classification and computer vision. In terms of spatial-temporal modeling of dynamical systems, the key relevant concepts of neural ODE are: (i) neural networks can be used to characterize the unknown vector field of a finite-dimensional dynamical system, and (ii) the unknown coefficients of the neural network can be trained via gradient descent using either the adjoint method or automatic differentiation. The idea of modeling continuous dynamics via machine learning methods was also discussed in [135]. In the past few years, several works [136–139] have explored the use of neural ODE in modeling dynamical systems and demonstrated promising results. Although neural ODE provides the flexibility of handling data with a non-uniform resolution in time, it has been shown in [140] that the use of a standard network in neural ODE can lead to long-term instability, mainly due to the amplification of high wavenumber components, when applying the trained network and simulating the modeled dynamical system. To address the long-term stability, the unknown vector field was modeled by linear and nonlinear parts with standard neural networks in [140] and the training via neural ODE demonstrated more stable long-term simulations. In this work, we show that the stable long-term simulations of neural ODE models can alternatively be achieved by filtering out some high wavenumber components through the construction of a neural dynamical operator, for which the Fourier neural operator [41] serves the purpose nicely.

Although neural ODE provides an efficient tool for training a model to match the short-term behavior of an unknown dynamical system, training the model to also quantitatively match the long-term system behavior still has some challenges. One challenge is on the computational side, that the long-term training would require huge memory costs [141–143] via backpropagation or potential stability issues in the backward-in-time simulation via the adjoint method. Another challenge is on the problem formulation side, that the long-term system behavior tends to be more sensitive to the small changes in the model, especially for chaotic systems or systems with discontinuities, making the standard adjoint method sometimes even infeasible [144]. These challenges motivate us to explore an alternative optimization approach that is both efficient and robust for matching the long-term behaviors between the model and the true dynamical system. More specifically, we choose the ensemble

Kalman inversion (EKI) method that was proposed in [68]. Unlike the backpropagation or the adjoint method that is designed for the gradient descent optimization, ensemble Kalman inversion is derivative-free, parallelizable, and robust with noises of the data and chaos or uncertainties of the system [66, 145]. In the past decade, many developments have been made to enhance Kalman inversion methods, both in theory [146–148] and in algorithms, such as various types of regularizations (e.g., linear equalities and inequalities [149], Tikhonov [150], sparsity [67], and ℓ_p [151], and known physics [152]), uncertainty quantification with Langevin dynamics for sampling [153], and using other types of Kalman filters [154]. More recently, the ensemble Kalman inversion was also explored in [155] for the training of neural ODE. Instead of merely using EKI to train a neural ODE, we investigate a hybrid optimization approach that uses the standard gradient-based method for short-term trajectory matching and the EKI method for long-term statistics matching.

In summary, we develop a spatially and temporally continuous framework called neural dynamical operator for the data-driven modeling of dynamical systems with spatial fields as system states, by leveraging the success of neural operator and neural ODE. Compared with classical methods, the proposed framework is more capable of capturing the underlying dynamics from data for both short-term state prediction and long-term statistics matching with resolution-invariant feature. The framework is tested with three spatiotemporal dynamical systems, including Burgers' equation, Navier–Stokes equation, and Kuramoto–Sivashinsky equation. The first two examples focus on the performances of neural dynamical operator trained with short-term data, to demonstrate the merits of the neural dynamical operator in terms of (i) spatial-temporal resolution-invariance and (ii) stable long-term simulations. The third example demonstrates the merit of a hybrid optimization method that leverages both short-term and long-term data to calibrate the neural dynamical operator. The key contributions of this work are summarized below:

- Combined Fourier neural operator and neural ODE to provide a spatial-temporal continuous framework for modeling unknown dynamical systems based on data in various resolutions.
- Demonstrated the long-term stable simulations of the trained models for three different spatiotemporal dynamical systems with discontinuous features or chaotic behaviors.
- Proposed a hybrid optimization scheme that efficiently leverages both the short-term time series and long-term statistics data.

2.2 Problem Statement

We focus on the continuous dynamical system in a general form:

$$\frac{\partial \mathbf{u}(\mathbf{x}, t)}{\partial t} = \mathcal{G}(\mathbf{u}(\mathbf{x}, t), t), \tag{2.1}$$

with spatial variable $\mathbf{x} \in D_{\mathbf{x}} \subseteq \mathbb{R}^{d_{\mathbf{x}}}$, temporal variable $t \in [0, T] \subset \mathbb{R}$, state function $\mathbf{u}(\cdot, \cdot) \in \mathcal{U}(D_{\mathbf{x}} \times [0, T]; \mathbb{R}^{d_{\mathbf{u}}})$, spatial profile of state function at time t is $\mathbf{u}(\cdot, t) \in \mathcal{U}_t(D_{\mathbf{x}}; \mathbb{R}^{d_{\mathbf{u}}})$, and $\mathcal{G} : \mathcal{U}_t \times [0, T] \mapsto \mathcal{U}_t$ is a non-linear operator. If the system is autonomous, i.e., the system does not depend on time, $\mathcal{G}$ would become a non-linear operator on spatial field. $\mathbb{R}^d$ is a real vector space with d dimension, $D_{\mathbf{x}}$ is a bounded domain, $\mathcal{U}$ and $\mathcal{U}_t$ are separable Banach spaces of functions taking values in $\mathbb{R}^{d_{\mathbf{u}}}$.

Assuming that the operator $\mathcal{G}$ in (2.1) is unknown, we propose to learn a data-driven dynamical operator $\tilde{\mathcal{G}}$ that approximates $\mathcal{G}$, based on the time series data of $\mathbf{u}$ and a combined use of neural ODE [38] and neural operator [41]. To enhance the performance of the learned model, the data-driven operator is trained to match both short-term trajectories and long-term statistics from the true system. A hybrid scheme of gradient-based and derivative-free methods is then demonstrated to facilitate efficient training with the combined types of data.

Given a time series $\{\mathbf{u}(t_n)\}_{n=0}^N$ of system state $\mathbf{u}$ in (2.1), where $\mathbf{u}(t) := \mathbf{u}(\cdot, t)|D_{\mathbf{x}}^{(M)} \in \mathbb{R}^{d_{\mathbf{u}} \times M}$ is discretization of $\mathbf{u}(\cdot, t)$ with $D_{\mathbf{x}}^{(M)} = \{x_1, ..., x_M\} \subset D_{\mathbf{x}}$ is a M-point discretization of the domain $D_{\mathbf{x}}$, we aim to build a continuous spatial-temporal model for the system by constructing a parametric map $\tilde{\mathcal{G}}$ with parameters $\boldsymbol{\theta}$ to approximate the non-linear operator $\mathcal{G}$. The continuous spatial-temporal model would then be:

$$\frac{\partial \tilde{\mathbf{u}}(\mathbf{x}, t)}{\partial t} = \tilde{\mathcal{G}}(\tilde{\mathbf{u}}(\mathbf{x}, t), t; \boldsymbol{\theta}). \tag{2.2}$$

To match with the time series data $\{\mathbf{u}(t_n)\}_{n=0}^N$, the unknown parameters $\boldsymbol{\theta}$ in $\tilde{\mathcal{G}}$ can be calibrated by minimizing the short-term loss for trajectory matching:

$$L_s := \sum_{n=0}^{N_s} J_s(\mathbf{u}(t_n),\ \tilde{\mathbf{u}}(t_n)), \tag{2.3}$$

where $\tilde{\mathbf{u}}(t_n) = \mathbf{u}(t_0) + \int_{t_0}^{t_n} \tilde{\mathcal{G}}(\tilde{\mathbf{u}}, t; \boldsymbol{\theta})dt$ is the predicted states, $\mathbf{u}(t_n)$ is the observed system states from the true system, and $J_s : \mathbb{R}^{d_{\mathbf{u}} \times M} \times \mathbb{R}^{d_{\mathbf{u}} \times M} \mapsto \mathbb{R}$ is a loss function at every time step. It is worth noting that the continuous counterpart of the total short-term loss L_s is a loss functional operating on two space-time continuous functions and is defined by a squared norm for the Bochner space $L^2([0, T_s]; L^2(D_{\mathbf{x}}; \mathbb{R}^{d_{\mathbf{u}}}))$, where $T_s = t_{N_s} - t_0$ corresponds to the total time range of short-term trajectory matching. The continuous counterpart of the loss function J_s is a functional on two spatial continuous functions which defined as $L^2(L^2(D_{\mathbf{x}}; \mathbb{R}^{d_{\mathbf{u}}}) \times L^2(D_{\mathbf{x}}; \mathbb{R}^{d_{\mathbf{u}}}); \mathbb{R})$. N_s is a hyper-parameter controlling the short-term state prediction horizon during training process. Given a long time series of states as data of a dynamical system, we sample a batch of short time series with length N_s from the dataset, with which we can use mini-batch training to obtain the parameters $\boldsymbol{\theta}$.

To capture a long-term statistical property of the true dynamical system, the unknown parameters $\boldsymbol{\theta}$ in $\tilde{\mathcal{G}}$ can be calibrated by minimizing the long-term loss for statistics matching:

$$L_l \left(\beta \left(\{\mathbf{u}(t_n)\}_{n=0}^{N_l} \right), \beta \left(\{\tilde{\mathbf{u}}(t_n)\}_{n=0}^{N_l} \right) \right), \tag{2.4}$$

where $\beta : \mathbb{R}^{d_{\mathbf{u}} \times M \times N_l} \mapsto \mathbb{R}^{d_\beta}$ maps a long-term trajectory to its important statistics, $L_l : \mathbb{R}^{d_\beta} \times \mathbb{R}^{d_\beta} \mapsto \mathbb{R}$ is a loss function to quantify the differences of long-term statistics between the modeled system and the true one, and N_l is another hyper-parameter that control the horizon over which the long-term statistics is calculated. The continuous counterpart of the map β operates on a space-time continuous function in the space $C([0, T_l]; L^2(D_{\mathbf{x}}; \mathbb{R}^{d_{\mathbf{u}}}))$, where $T_l = t_{N_l} - t_0$ corresponds to the total time range of long-term statistics matching. In practice, to reduce the computational costs during the training process, we sample a batch of relatively long time series with length N_l instead of calculating the long-term statistics for the entire time series dataset. The long-term horizon is typically much larger than the short-term horizon, i.e., $T_l \gg T_s$.

2.3 Methodological Background

2.3.1 Neural Operator

To construct the data-driven dynamical operator $\tilde{\mathcal{G}}$ in the continuous setting, i.e., viewing the system state $\mathbf{u}$ as a continuous spatial-temporal field instead of a discrete finite-dimensional

vector, we rely on the recent developments of neural operators [40, 41]. As a high-level framework for modeling continuous dynamical systems, the neural dynamical operator can utilize any neural operator architecture to construct the data-driven dynamical operator $\tilde{\mathcal{G}}$. In this work, we present results using Fourier neural operator (FNO) [41]. In general, neural operators can approximate $\mathcal{G}$ in (2.1) which is a non-linear mapping between infinite-dimensional spaces:

$$\mathcal{G} : \mathcal{U}_t \times [0, T] \mapsto \mathcal{U}_t, \tag{2.5}$$

with a neural network $\tilde{\mathcal{G}}(\cdot;\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$. The key advantages of continuous mapping in (2.5) are: (i) the performance of the trained mapping is resolution-invariant, and (ii) the flexibility of using data with different discretizations. The resolution-invariant feature allows the model to learn from and evaluate the functions discretized in an arbitrary way. In other words, the model trained at one specific resolution can be used to perform inference at different resolutions of input and output functions, without a significantly increased error level. With the resolution-invariant property in the construction of the FNO architecture, the training process can also potentially combine the use of data in different resolutions.

We then focus on the use of FNO to construct the data-driven operator $\tilde{\mathcal{G}}$. The FNO applies a Fourier transform to convert the input spatial function from the spatial domain to the Fourier domain, processes the data within the Fourier domain, and subsequently performs an inverse Fourier transform to transform the results back to the spatial domain. Similarly, when the input function is defined on a temporal domain, it can still be transformed between the temporal domain and the Fourier domain. More specifically, FNO first linearly transforms the state function $\mathbf{u}(\cdot, t) \subset \mathcal{U}_t$ and temporal variable t to lift the dimension from $\mathbb{R}^{d_u+1}$ to $\mathbb{R}^{d_v}$: $\mathbf{v}_0(\mathbf{x}) = P(\mathbf{u}(\mathbf{x}, t), t)$. Then $\mathbf{v}_0(\mathbf{x})$ will be iteratively transformed by Fourier layers with the output dimension staying the same as $d_{\mathbf{v}}$:

$$\mathbf{v}_{n+1}(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{v}_n(\mathbf{x}) + (\mathbf{K}\mathbf{v}_n)(\mathbf{x})), \quad n = 0, 1 \dots N-1, \tag{2.6}$$

where $(\mathbf{K}\mathbf{v}_n)(\mathbf{x}) = \mathcal{F}^{-1}(\mathbf{R} \cdot (\mathcal{F}\mathbf{v}_n))(\mathbf{x})$ is the Fourier integral operator, $\mathcal{F}$ is Fourier transform and $\mathcal{F}^{-1}$ is its inverse, $\mathbf{R}$ is the complex-valued parameters in the neural network, $\mathbf{W}$ is a linear transformation, and σ is a nonlinear activation function. At last, $\mathbf{v}_N(\mathbf{x})$ will be linearly transformed again to ensure the dimension of the final output same as the dynamics function in the space $\mathcal{U}_t$, i.e., $Q(\mathbf{v}_N(\mathbf{x})) = \frac{\partial \tilde{\mathbf{u}}(\mathbf{x},t)}{\partial t} \in \mathbb{R}^{d_u}$.

The classical loss function for training FNO quantifies the differences between the true dynamics $\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t}$ and the predicted dynamics $\frac{\partial \tilde{\mathbf{u}}(\mathbf{x},t)}{\partial t}$. However, the data of $\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t}$ may not be available in many applications, e.g., when the temporal resolution in the data of $\mathbf{u}(\mathbf{x},t)$ is not fine enough to obtain an informative estimation of its time derivative. Therefore, we may not be able to directly train the neural dynamical operator $\tilde{\mathcal{G}}$ by solving the optimization problem as typically done in the classical usage of FNO. To address this issue, we explore the training of a neural dynamical operator via the framework of neural ODE [38].

2.3.2 Neural Ordinary Differential Equations

The goal of neural ordinary differential equations (neural ODE) is to train the surrogate model $\tilde{\mathcal{G}}$, which is constructed via FNO in this work, to approximate $\mathcal{G}$ in (2.1) based on time series data $\{\mathbf{u}(t_n)\}_{n=0}^{N}$ with $\mathbf{u}(t) := \mathbf{u}(\cdot,t)|D_{\mathbf{x}}^{(M)}$. With a given $\tilde{\mathcal{G}}$ and initial state of true system $\mathbf{u}(t_0)$, the predictive system state at future time t_n can be written as $\tilde{\mathbf{u}}(t_n) = \mathbf{u}(t_0) + \int_{t_0}^{t_n} \tilde{\mathcal{G}}(\tilde{\mathbf{u}}(t),t;\boldsymbol{\theta})dt$, which can be obtained by an ODE solver in real applications. The trainable parameters $\boldsymbol{\theta}$ in $\tilde{\mathcal{G}}$ can then be obtained by minimizing the loss function in (2.3). With the loss function J_s at each time step chosen as standard ℓ^2-norm in this work, the total short-term loss for trajectory matching becomes

$$L_s = \sum_{n=0}^{N_s} \|\mathbf{u}(t_n) - \tilde{\mathbf{u}}(t_n)\|^2. \tag{2.7}$$

The neural ODE framework [38] discussed two methods to calculate $dL_s/d\boldsymbol{\theta}$, including (i) backpropagation and (ii) adjoint sensitivity method, with which we can determine the optimal $\boldsymbol{\theta}$ by minimizing the short-term loss in (2.7) via gradient descent.

Backpropagation is a classical gradient descent optimization method for training a neural network. One drawback of the backpropagation for neural ODE training is the memory cost to store the results of its forward pass, which is linearly proportional to the number of $\tilde{\mathcal{G}}$ evaluations. Although various types of memory management techniques [143] have been developed for automatic differentiation, the memory cost for matching a long time series data with a large scale model of $\tilde{\mathcal{G}}$ can still be inefficient or even infeasible.

On the other hand, the adjoint sensitivity method calculates the gradient $dL_s/d\boldsymbol{\theta}$ by a forward and a backward ODE integration. The forward integration solves for the state $\tilde{\mathbf{u}}$ at future time step $t_1, t_2, \ldots, t_N$, with the initial state $\tilde{\mathbf{u}}(t_0) = \mathbf{u}(t_0)$. By introducing

an adjoint state $\mathbf{a}(t_n) = \frac{\partial L_s}{\partial \tilde{\mathbf{u}}(t_n)}$, the backward integration solves for an augmented states $[\tilde{\mathbf{u}}(t), \mathbf{a}(t), \frac{\partial L_s}{\partial \boldsymbol{\theta}}]^\top$ from t_n to t_{n-1}, for $n = N, N-1, \ldots, 1$, with the initial state $\tilde{\mathbf{u}}(t_N)$ evaluated from the forward integration, $\mathbf{a}(t_N) = \frac{\partial L_s}{\partial \tilde{\mathbf{u}}(t_N)}$, and $\frac{\partial L_s}{\partial \boldsymbol{\theta}}(t_N) = \mathbf{0}$:

$$\begin{aligned} \frac{d\tilde{\mathbf{u}}(t)}{dt} &= \tilde{\mathcal{G}}(\tilde{\mathbf{u}}(t), t; \boldsymbol{\theta}), \\ \frac{d\mathbf{a}(t)}{dt} &= -\mathbf{a}(t)^\top \frac{\partial \tilde{\mathcal{G}}(\tilde{\mathbf{u}}(t), t; \boldsymbol{\theta})}{\partial \tilde{\mathbf{u}}(t)}, \\ \frac{d}{dt}\frac{\partial L_s}{\partial \theta} &= -\mathbf{a}(t)^\top \frac{\partial \tilde{\mathcal{G}}(\tilde{\mathbf{u}}(t), t; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \end{aligned} \tag{2.8}$$

It should be noted that the solved adjoint state $\mathbf{a}(t_n)$ needs to be adjusted at $n = N-1, N-2, \ldots, 0$ during the backward integration, by adding a term of $\frac{\partial L_s}{\partial \tilde{\mathbf{u}}(t_n)}$ to account for the fact that the loss function L_s explicitly depends on the system state $\tilde{\mathbf{u}}(t_n)$. During the whole backward integration, the vector-Jacobian products $\mathbf{a}^\top \frac{\partial \tilde{\mathcal{G}}}{\partial \tilde{\mathbf{u}}}$ and $\mathbf{a}^\top \frac{\partial \tilde{\mathcal{G}}}{\partial \boldsymbol{\theta}}$ can be evaluated on the fly without storing them in the memory. Therefore, the adjoint sensitivity method has a constant memory usage $\mathcal{O}(1)$ with respect to the integration time steps, i.e., the number of $\tilde{\mathcal{G}}$ evaluations.

For a short-term integration, the backpropagation often has no memory cost issue and is computationally faster than the adjoint method, considering that the Jacobian is stored and does not need to be evaluated on the fly in the backward pass. However, the memory cost issue would prevent the use of backpropagation if the data involves long-term integration, e.g., time-averaged statistics. In addition, gradient-based optimization can potentially encounter numerical issues (e.g., gradient blow-up) for the long-term information of chaotic systems. Therefore, backpropagation is used in this paper for short-term trajectory matching, while the long-term statistics of a chaotic dynamical system are incorporated by a derivative-free Kalman method, instead of the adjoint sensitivity method. More details about the derivative-free Kalman method can be found in Section 2.3.3.

2.3.3 Ensemble Kalman Inversion

The neural dynamical operator $\tilde{\mathcal{G}}$ in (2.2) can be calibrated to capture the long-term statistics of the true dynamical system by solving the optimization problem in (2.4) with ensemble Kalman inversion (EKI). Originated from ensemble Kalman filter [71, 73, 96], EKI [68, 97]

has been developed as an optimization method to solve inverse problems. More specifically, the goal of EKI is to estimate the unknown parameters from noisy data in the general form of an inverse problem:

$$\mathbf{y} = G(\boldsymbol{\theta}) + \boldsymbol{\eta}, \tag{2.9}$$

where $\mathbf{y}$ is a vector of observation data, G denotes a forward map, $\boldsymbol{\theta}$ corresponds to the trainable parameters, $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\eta}})$ is often chosen as Gaussian random noises with a covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$. In this work, $\mathbf{y} = \beta(\{\mathbf{u}(t_n)\}_{n=0}^{N_l}) \in \mathbb{R}^{d_\beta}$ is the observed long-term statistics of the dynamical system, and the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\eta}} \in \mathbb{R}^{d_\beta \times d_\beta}$ is empirically chosen. In practice, the covariance matrix can also be determined by observing an ensemble of long-term statistics from the true system, with each ensemble member having a different initial condition. The forward map G is a composition of data-driven dynamical operator $\tilde{\mathcal{G}}$, a time integral operator $\int$ that is numerically evaluated by an ODE solver, and β that calculates the time-averaged statistics from a time series of system states, i.e.,

$$G(\boldsymbol{\theta}) := \beta(\{\tilde{\mathbf{u}}(t_n)\}_{n=0}^{N_l}) = \beta\left(\left\{\mathbf{u}(t_0) + \int_{t_0}^{t_n} \tilde{\mathcal{G}}(\tilde{\mathbf{u}}(\mathbf{x}, t), t; \boldsymbol{\theta}) dt\right\}_{n=0}^{N_l}\right). \tag{2.10}$$

To match the observed data $\mathbf{y}$ and the output of forward map G, EKI is employed to identify the optimal parameters $\boldsymbol{\theta}$. The optimization problem in (2.4) solved by EKI can be written as

$$\min_{\boldsymbol{\theta}} ||\boldsymbol{\Sigma}_{\boldsymbol{\eta}}^{-\frac{1}{2}}(\mathbf{y} - G(\boldsymbol{\theta}))||^2. \tag{2.11}$$

To solve the optimization problem in (2.11), the EKI formula that iteratively updates the ensemble estimation of parameters $\{\boldsymbol{\theta}^{(j)}\}_{j=1}^{J}$ is:

$$\boldsymbol{\theta}_{n+1}^{(j)} = \boldsymbol{\theta}_n^{(j)} + \boldsymbol{\Sigma}_n^{\boldsymbol{\theta}\mathbf{g}}(\boldsymbol{\Sigma}_n^{\mathbf{gg}} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}})^{-1}(\mathbf{y}^{(j)} - \mathbf{g}_n^{(j)}), \tag{2.12}$$

where the index n denotes the n-th EKI step and $\mathbf{y}^{(j)}$ corresponds to the perturbed observation data $\mathbf{y}$ via sampling the noises $\boldsymbol{\eta}$. With the ensemble of parameters $\{\boldsymbol{\theta}_n^{(j)}\}_{j=1}^{J}$ at the n-th

EKI step, the terms $\mathbf{g}_n^{(j)}$, $\boldsymbol{\Sigma}_n^{\boldsymbol{\theta g}}$, and $\boldsymbol{\Sigma}_n^{\mathbf{gg}}$ in (2.12) are calculated as:

$$\begin{aligned}
\bar{\boldsymbol{\theta}}_n = \frac{1}{J}\sum_{j=1}^{J}\boldsymbol{\theta}_n^{(j)},\ \ \mathbf{g}_n^{(j)} = G(\boldsymbol{\theta}_n^{(j)}),\ \ \bar{\mathbf{g}}_n = \frac{1}{J}\sum_{j=1}^{J}\mathbf{g}_n^{(j)},\\
\boldsymbol{\Sigma}_n^{\boldsymbol{\theta g}} = \frac{1}{J-1}\sum_{j=1}^{J}(\boldsymbol{\theta}_n^{(j)} - \bar{\boldsymbol{\theta}}_n)(\mathbf{g}_n^{(j)} - \bar{\mathbf{g}}_n)^\top,\\
\boldsymbol{\Sigma}_n^{\mathbf{gg}} = \frac{1}{J-1}\sum_{j=1}^{J}(\mathbf{g}_n^{(j)} - \bar{\mathbf{g}}_n)(\mathbf{g}_n^{(j)} - \bar{\mathbf{g}}_n)^\top.
\end{aligned} \tag{2.13}$$

2.4 Neural Dynamical Operator

Based on the techniques introduced in Sections 2.3.1 and 2.3.2, we present a spatial-temporal continuous framework that learns a data-driven dynamical operator $\tilde{\mathcal{G}}$ in (2.2) based on the short-term time series of the true system states. More specifically, the key components of the proposed framework include:

- Constructing the dynamical operator $\tilde{\mathcal{G}}$ via a Fourier neural operator.
- With the short-term time series of the true system states, updating the parameters θ of the dynamical operator $\tilde{\mathcal{G}}$ via solving the optimization problem in (2.3) based on a gradient-based optimization method (e.g., neural ODE).

The merits of learning a dynamical operator $\tilde{\mathcal{G}}$ in (2.2) are:

- The flexibility of using non-uniform data points in both space and time.
- The resolution-invariance of the trained model in both space and time.

Besides being good at predicting short-term system states, the trained modeled system in (2.2) also demonstrates a stable long-term simulation. There are some existing methods to make stable long-term predictions for chaotic dynamical systems such as reduced order modeling [136] and stabilized neural ODE [140]. Instead of preventing the high-wavenumbers amplification by dimension reduction or employing a linear damping term, the use of the Fourier neural operator in this work facilitates the stable long-term simulation by filtering out the high-order modes in the Fourier space. Therefore, neural dynamical operator trained

via short-term time series data can still avoid a numerical explosion in long-term simulations and also qualitatively retain the statistical property of the true system.

However, the use of short-term time series data alone could not guarantee a quantitative match of the long-term statistics between the trained model and the true system. This limitation motivates the combined use of short-term time series and long-term time-averaged statistics as two types of data for training the neural dynamical operator. A hybrid optimization scheme that can efficiently incorporate both types of data is introduced in Section 2.5.

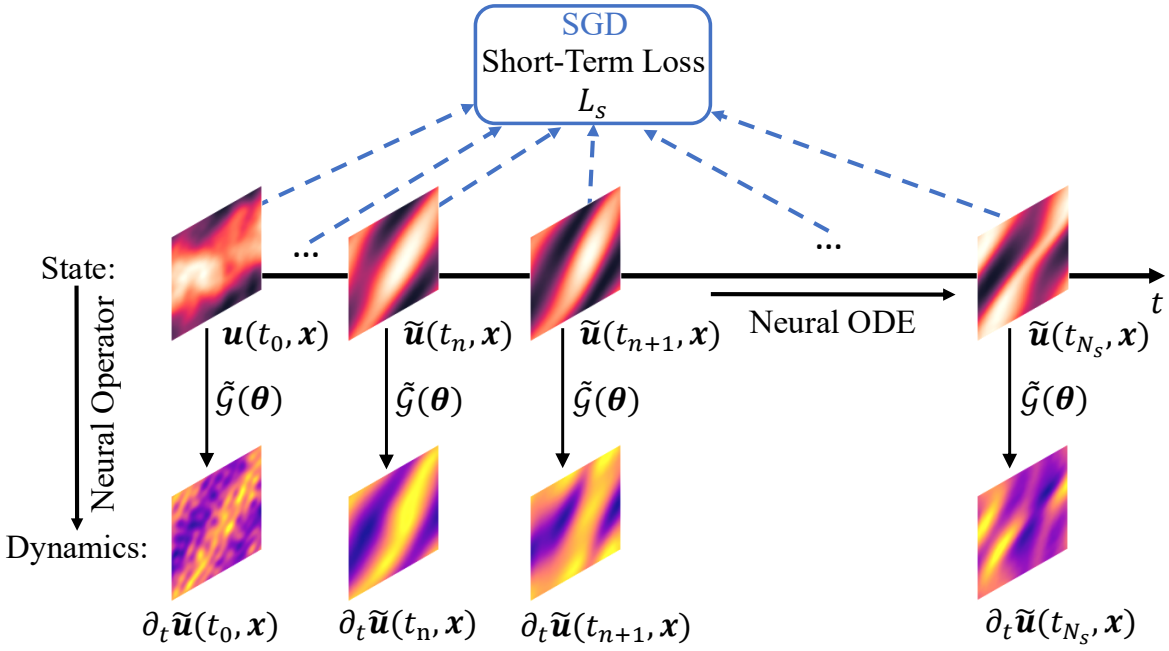

Figure 2.1: Schematic diagram of neural dynamical operator (based on Navier–Stokes equations). The dynamics of the system for the current state are approximated by neural operator, then the future states are evaluated with an ODE solver along with time given any initial state. The neural dynamical operator $\tilde{\mathcal{G}}$ is trained by minimizing the Loss L_s with gradient-based optimization.

2.5 A Hybrid Optimization Scheme: Integrating Gradient-Based and Derivative-Free Methods

To efficiently incorporate both short-term time series and long-term statistics of the true system states as the training data, we propose a hybrid optimization scheme that iteratively solves the optimization problems in (2.3) and (2.4). Within each iteration, the parameters $\boldsymbol{\theta}$ of the dynamical operator $\tilde{\mathcal{G}}$ in (2.2) are first updated via a gradient-based optimization method (e.g., backpropagation) that solves (2.3) with the short-term time series data of the true system states and then further adjusted via a derivative-free optimization method (e.g., ensemble Kalman inversion) to account for the long-term statistics data in (2.4). Based on the history of both short-term state error and long-term statistics error during the EKI updating, we select a robust $\boldsymbol{\theta}$ that yields relatively small values for both types of errors. From a more general perspective, the hybrid optimization aims to solve a dual-objective optimization problem: it seeks to minimize both the loss from short-term state prediction and the loss from long-term statistics matching. The optimization procedure leads to a set of candidate solutions that approximates the Pareto front, from which we can robustly select the solution that achieves a balanced performance of predicting the short-term trajectory and long-term statistics. The Algorithm 1 briefly summarizes the hybrid optimization scheme, while a detailed version is presented in Algorithm 3 in A.

Algorithm 1 Training neural dynamical operator with the hybrid optimization scheme (A brief version)

for $n = 1$ **to** N_{hybrid} **do**
  for $n_1 = 1$ **to** N_{SGD} **do**
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} L_s$ ▷ SGD
  end for
  $\{\boldsymbol{\theta}^{(j)}\}_{j=1}^{J} \leftarrow \boldsymbol{\theta} + \{\boldsymbol{\epsilon}^{(j)}\}_{j=1}^{J}$ ▷ Initialize ensemble
  for $n_2 = 1$ **to** N_{EKI} **do**
    $\left\{\boldsymbol{\theta}^{(j)}\right\}_{j=1}^{J} \leftarrow \{\boldsymbol{\theta}^{(j)} + \boldsymbol{\Sigma}^{\boldsymbol{\theta g}}(\boldsymbol{\Sigma}^{\mathbf{gg}} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}})^{-1}(\mathbf{y}^{(j)} - \mathbf{g}^{(j)})\}_{j=1}^{J}$ ▷ EKI
  end for
  $\boldsymbol{\theta} \leftarrow \frac{1}{J}\sum_{j=1}^{J} \boldsymbol{\theta}^{(j)}$ ▷ Parameter mean
end for
return robust $\boldsymbol{\theta}^*$ based on the error history during EKI updating

The key merit of the hybrid optimization scheme is the efficient incorporation of both

short-term time series and long-term time-averaged statistics data. With the use of both types of data, the trained neural dynamical operator is expected to have better generalization capability, which is confirmed by the numerical example of the Kuramoto—Sivashinsky equation in Section 2.6.3. Though the two optimization processes are independent of each other, the selection of the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$ in (2.11) will still affect results of hybrid optimization. The reason is that the same parameters in the neural dynamical operator are updated by both optimization process. If the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$ is of a small magnitude, i.e., the observation noises of long-term statistics are small, the hybrid optimization would prefer smaller mismatches in long-term statistics from the true system, which could negatively impact the performance of the hybrid optimization, especially when the finite time-averaged approximation of the long-term statistics have larger intrinsic uncertainties compared to the level of observation noises in EKI. The intrinsic uncertainties due to the finite time-averaged approximation of the long-term statistics can be estimated by obtaining an ensemble observation of long-term statistics, which can be used to set up the observation noises of long-term statistics in EKI. In practice, a simpler choice for $\boldsymbol{\Sigma}_{\boldsymbol{\eta}}$ is a diagonal matrix, e.g., $c^2\mathbf{I}_{d_y}$ where $c \in \mathbb{R}$ is a constant scalar and $\mathbf{I}_{d_y} \in \mathbb{R}^{d_y \times d_y}$ is an identity matrix with dimension $d_{\mathbf{y}}$. With this choice, the scalar c plays the role of the hyper-parameter that controls the results of hybrid optimization. In this work, the constant c is empirically chosen as 0.1, based on the magnitude of intrinsic uncertainties due to finite time-averaged approximation of long-term statistics estimated by ensemble simulations. Alternatively, the hyper-parameter c can be determined by cross-validation, i.e., choosing the optimal c that provides the best cross-validation results.

2.6 Numerical Experiments

We demonstrate the performance of the continuous spatial-temporal model on three examples, including (i) 1-D viscous Burgers' equation, (ii) 2-D Navier–Stokes equations, and (iii) Kuramoto–Sivashinsky equation. Short-term time series data generated from each true system are sub-sampled in both spatial and temporal domain with various resolutions, to confirm the resolution-invariance of the trained model with respect to both spatial and temporal discretizations. For all the examples, we also show the stable long-term simulation with the trained models, which is mainly due to the high-wavenumber filtering in the Fourier neural operator. For the example of Kuramoto–Sivashinsky equation, we

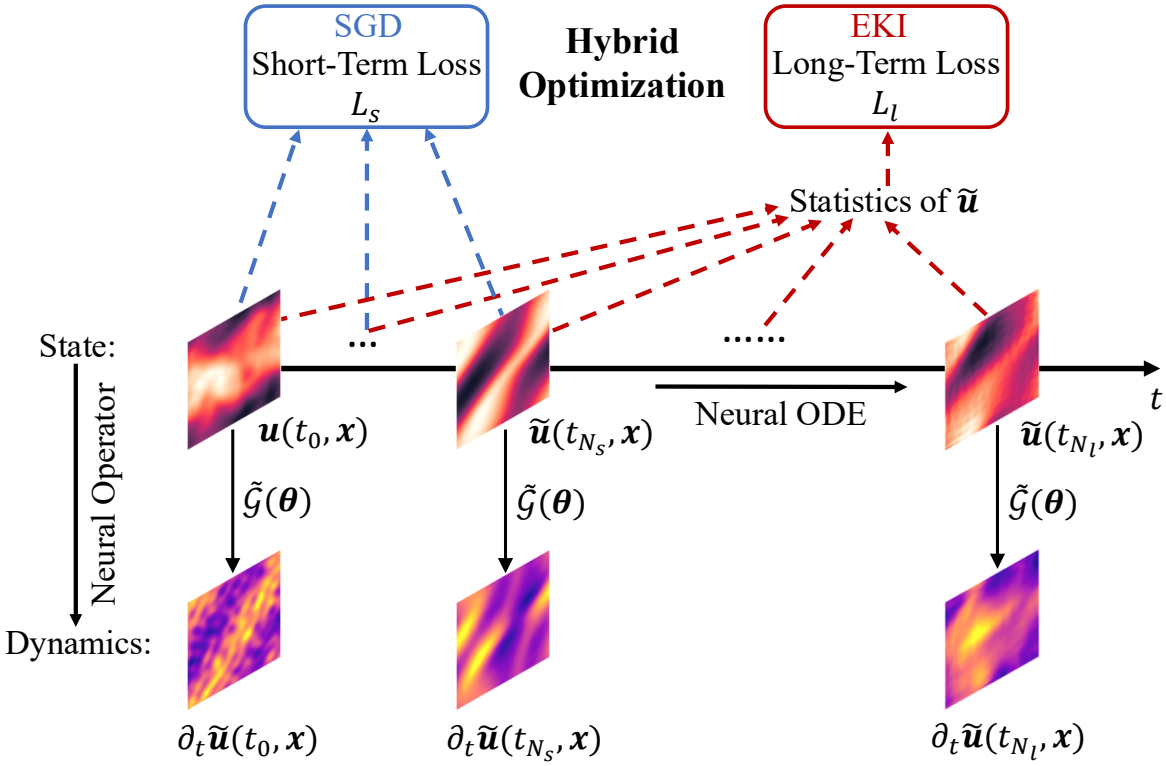

Figure 2.2: Schematic diagram of neural dynamical operator with hybrid optimization scheme (based on Navier–Stokes equations). To better generalize the model by utilizing both short-term and long-term data, the neural dynamical operator $\tilde{\mathcal{G}}$ is trained by the hybrid optimization scheme which will iteratively update parameters by stochastic gradient descent (SGD) method to minimize short-term states loss $\mathrm{L_s}$ and by derivative-free method (EKI) to minimize long-term statistics loss $\mathrm{L_l}$. The short-term system evolution in $[\mathrm{t_0}, \mathrm{t_{N_s}}]$ corresponds to Figure 2.1.

present a quantitative comparison of the long-term statistics between the model trained with short-term time series data and the one trained with both short-term time series and long-term statistics data. The results demonstrate the merit of the combined use of data via the proposed hybrid optimization scheme. For all the examples, the fixed-step Runge-Kutta method (RK4) is used as the ODE solver, the absolute error is the mean squared error, and the relative error is calculated by the average of $\frac{\|\mathbf{u}-\tilde{\mathbf{u}}\|_2}{\|\mathbf{u}\|_2}$ across sample size where $\|\cdot\|_2$ is ℓ^2-norm.

2.6.1 Viscous Burgers' Equation

Burgers' equation is a classical example of partial differential equation and has been widely used in many fields such as fluid mechanics [156–158], nonlinear dynamics [159, 160] and traffic flow [161, 162]. The governing equation of the viscous Burgers' equation is:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}, \tag{2.14}$$

where $x \in (0, L_x)$ with periodic boundary conditions, $t \in (0, L_t]$, ν is the viscosity coefficient, and $u(x, 0)$ is a given initial condition. Neural dynamical operator aim to learn the operator on the right-hand-side of (2.14), i.e., $\mathcal{G} : u \mapsto \frac{\partial u}{\partial t}$. By studying this example, we demonstrate that the trained neural dynamical operator has resolution-invariance with respect to both spatial and temporal discretizations. Also, the trained model can capture the shock behavior and shows a good performance in predicting the trajectory of the true system in the test dataset.

The simulation settings are $L_x = 1, L_t = 5, \nu = 10^{-3}, \Delta x = 1/1024, \Delta t = 0.005$. We perform 1000 simulations as training data and another 100 simulations as test data. The initial conditions of those simulations are randomly sampled from a 1-D Gaussian random field $\mathcal{N}(0, 625(-\Delta + 25I)^{-2})$ with periodic boundary conditions, where Δ is a Laplace operator and I is an identity operator.

In the optimization problem in (2.3), we first simulate the modeled system for $L_t = 5$ time units given an initial state $u(t_0)$ to obtain the trajectory $\{\tilde{u}(t_n)\}_{n=1}^{N_s}$ with $t_{N_s} = 5$ and then minimize the ℓ^2-norm between the simulated system trajectory and the true one. During the training process, we select a small subset from 1000 training simulations as a mini-batch to perform the gradient descent optimization in each epoch.

To construct the neural dynamical operator $\tilde{\mathcal{G}}$, we use FNO as a surrogate model with $d_v = 64$ and $k_{max} = 24$. d_v is a higher dimension where the input data be lifted to and k_{max} is a cut point above which the modes of Fourier transform of input data will be truncated. We train the model with 10^3 epochs with 10 simulations as one data batch in each epoch. The optimizer is Adam with 10^{-3} learning rate and cosine annealing schedule.

We train the neural dynamical operator based on short-term time series data with various resolutions in both space and time. The test errors are summarized in Table 2.1. To demonstrate the resolution-invariance of the trained models in both space and time, we present two types of test errors in Table 2.1: the Test Error (I) is based on the test data of the

same resolution $\Delta x = 1/1024$ and $\Delta t = 0.05$, and the Test Error (II) is based on resolution setting same as each train data. By comparing these two types of test errors with a fixed training resolution, it can be seen that the test error stays at the same order of magnitude when predicting on a finer resolution test dataset. On the other hand, the comparison among the cases with different training data resolutions demonstrates that the trained model can still perform well with a relatively sparse temporal resolution. The resolution-invariance property of the trained neural dynamical operator makes it flexible in using training data with low or even mixed resolutions.

Table 2.1: The test errors of viscous Burgers' equation with various resolution settings for train data. The Test Error (I) is based on the test data of the resolution $\Delta x = 1/1024$ and $\Delta t = 0.05$, while the Test Error (II) is based on a resolution setting the same as each training data.

Error / Resolution	Train Data		Test Error (I)		Test Error (II)	
	Δx	Δt	Absolute	Relative	Absolute	Relative
Resolution1	1/512	0.05	6.7010e-05	4.4004e-02	6.7595e-05	4.4163e-02
Resolution2	1/256	0.1	6.6947e-05	4.3981e-02	6.8292e-05	4.4299e-02
Resolution3	1/64	0.5	7.7509e-05	4.7405e-02	7.3959e-05	4.7423e-03

The Burgers' equation can develop shocks over time in the absence of viscous term. In (2.14), the non-linear convective term $u\frac{\partial u}{\partial x}$ can result in a shock in the solution of u (i.e., discontinuity of u in space) when $\nu = 0$. With $\nu \neq 0$, the diffusion term $\nu\frac{\partial^2 u}{\partial x^2}$ will lead to a continuous solution of u, while the spatial gradient of u can be large at certain locations if ν is small. In this example, we choose a relatively small value of viscosity, i.e., $\nu = 10^{-3}$ so that the solutions of the true system have such a feature of large spatial gradients. The results in Figures 2.3 and 2.4 confirm that the trained models can capture this behavior and provide good performance for short-term trajectory prediction of the solutions of u.

In Figure 2.3, we present the spatial-temporal plots of the solutions from the true system and the modeled ones, with the initial condition sampled from test data. The left column presents the true solution of u. The middle column corresponds to the solutions from the modeled systems trained with different resolution settings in Table. 2.1, and we test the prediction performance of these trained models with the same resolution setting (i.e., $\Delta x = 1/1024$ and $\Delta t = 0.05$, which is finer than all the training resolutions). The left column shows the differences between the true solution and the solutions from the modeled systems.

We can see that all predictions (shown in the middle column in Figure 2.3) capture the overall pattern of the true system. On the other hand, the absolute errors made by those predictions show that all trained models can provide relatively small errors at most locations and times, even testing on a finer resolution and starting with an unseen initial condition. It should be noted that the absolute errors tend to be larger close to the regions where the true solution has a large spatial gradient, but the magnitudes of those errors are still small compared to the true solution in those regions.

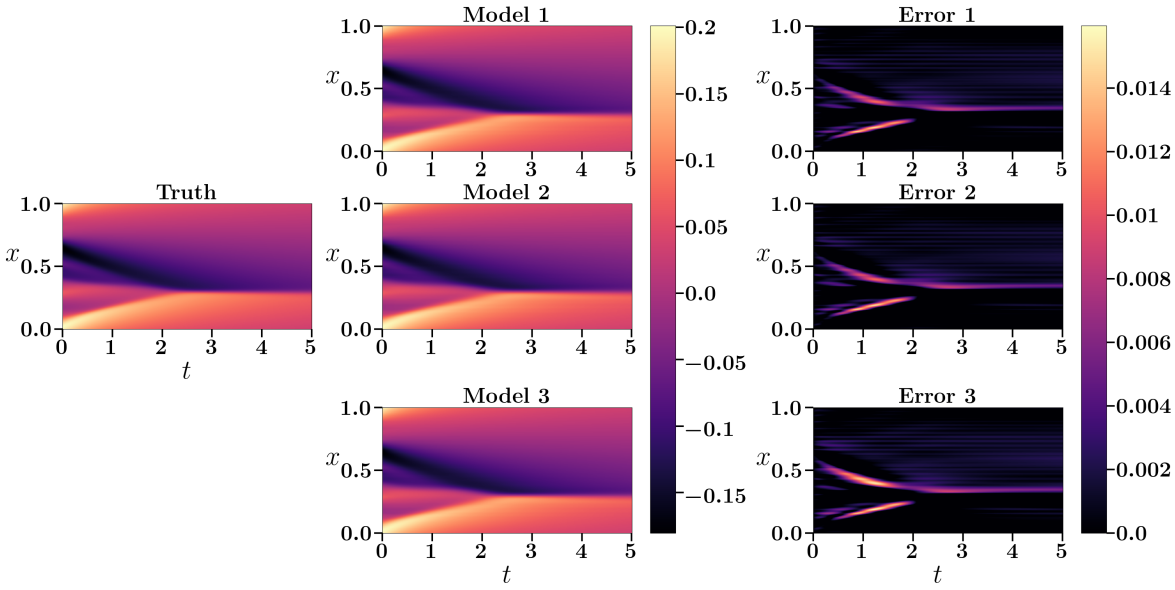

Figure 2.3: The spatial-temporal solutions of viscous Burgers' equation. Left column: true system. Middle column: trained models from three different resolutions with the same test data resolution ($\Delta x = 1/1024, \Delta t = 0.05$). Right column: errors of the solutions simulated based on the trained models. The index in the three trained models corresponds to the resolution settings in Table 2.1.

We also study the comparison of the energy spectrum between the true system and the model ones. The energy spectrum is defined as:

$$E(k,t) = \frac{1}{2}|u_f(k,t)|^2, \tag{2.15}$$

where $u_f = \mathcal{F}u$ denotes the Fourier transform of the solution u with k as the wavenumber. Here $|\cdot|$ evaluates the magnitude of a complex number.

Figure 2.5 presents the energy spectrum of the solution u for the true system and the

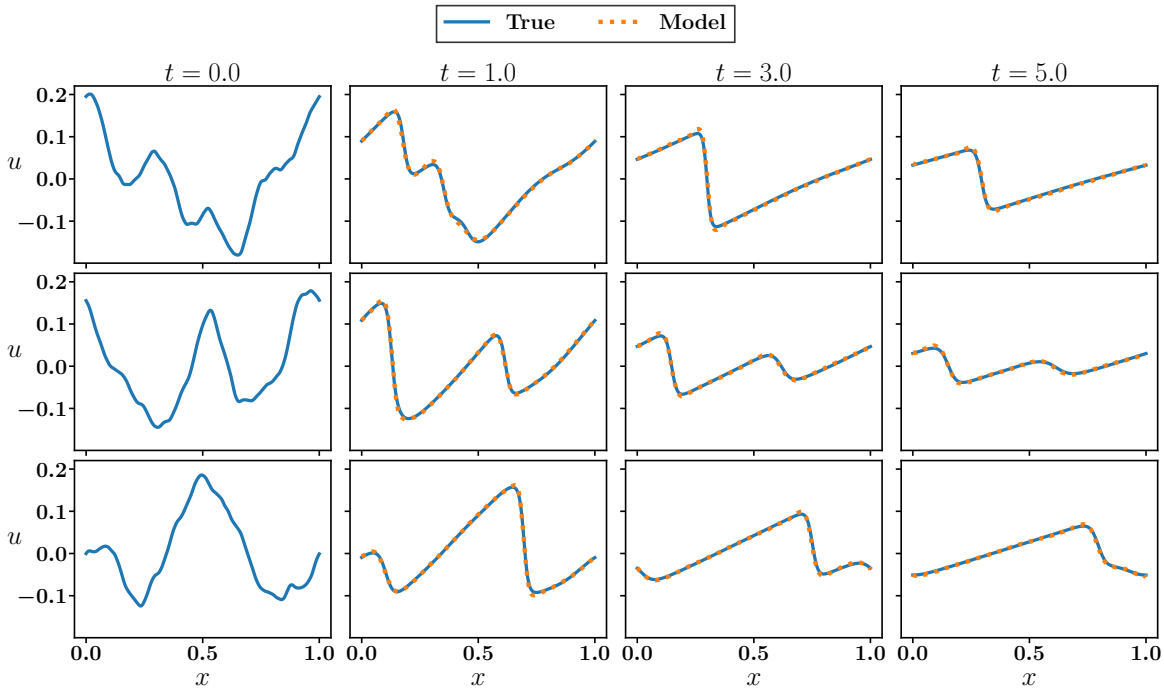

Figure 2.4: Solution profiles of viscous Burgers' equation for the true system and the model ones with different initial conditions from test data. The model is trained in a coarse resolution ($\Delta x = 1/64, \Delta t = 0.5$) and tested on a finer resolution ($\Delta x = 1/1024, \Delta t = 0.05$).

model ones. The spectrum shows a slope k^{-2}, which agrees well with [140]. The three trained models are tested with the same resolution ($\Delta x = 1/1024$, $\Delta t = 0.05$). It can be seen that the energy spectrum of the trained models has good agreement with each other, indicating that the resolution-invariance is achieved by the proposed neural dynamical operator. In addition, we can see that the trained models can capture the true spectrum up to a wavenumber of approximately 200 to 300. It is promising to see that the trained models can generalize quite well with all these unseen initial conditions in most wavenumbers and only start to display mismatches for very high wavenumbers.

2.6.2 Navier–Stokes Equations

We consider the Navier–Stokes equations to study the performance of neural dynamical operator on a 2-D continuous dynamical system. The Navier–Stokes equations are partial differential equations that characterize the conservation of linear momentum in fluid flows [163–166]. The 2-D Navier–Stokes equations written in the form of vorticity are:

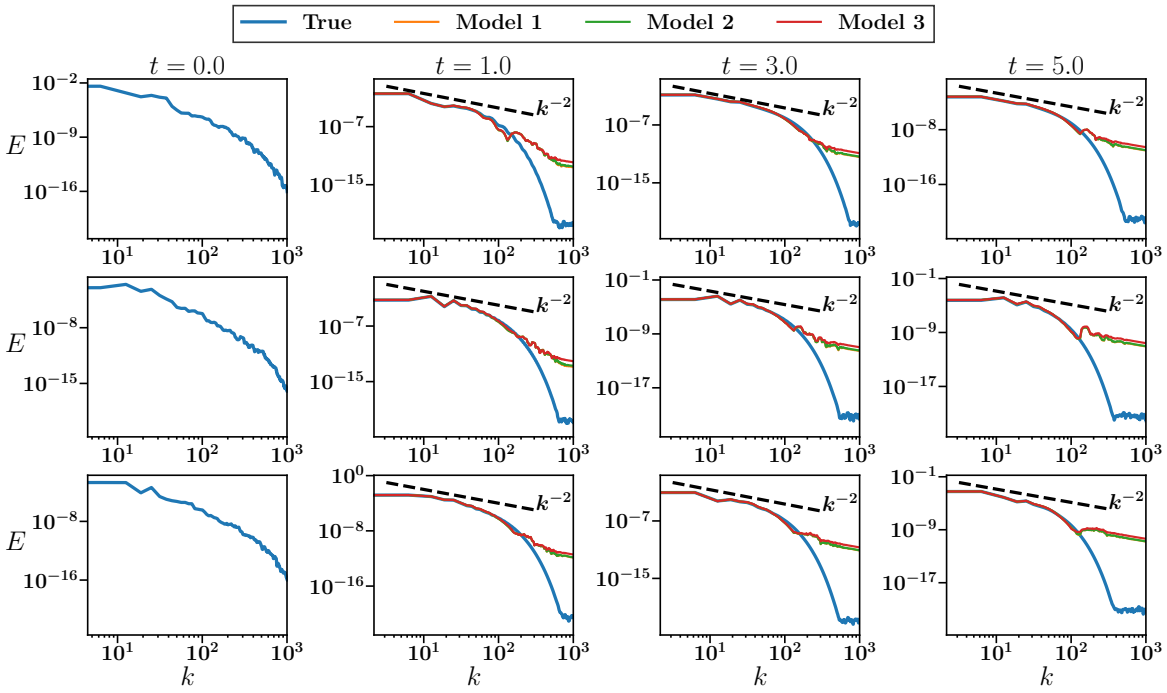

Figure 2.5: Energy spectrum of the true system and the model ones with different test initial conditions (in rows) and at different times (in columns). The index in the three trained models corresponds to the resolution settings in Table 2.1.

$$\begin{aligned} \frac{\partial \omega}{\partial t} &= -u \cdot \nabla \omega + \nu \Delta \omega + f, \\ \nabla \cdot u &= 0, \end{aligned} \tag{2.16}$$

where u denotes a 2-D velocity vector, $\omega := \nabla \times u$ represents the vorticity, ν is the kinematic viscosity of the fluid, and f corresponds to a forcing function. Neural dynamical operator aim to learn the operator on the right-hand-side of (2.16), i.e., $\mathcal{G} : \omega \mapsto \frac{\partial \omega}{\partial t}$. In real applications, the training data generated by simulations may not be with enough high resolution to well capture the true operator $\mathcal{G}$, e.g., high Reynolds number wall-bounded turbulent flows, for which resolving the Kolmogorov scales is still infeasible for many real engineering applications. With this example, we demonstrate that the trained neural dynamical operator can still capture the resolved information from a dataset, even with relatively low spatial resolution. However, the trained operator may not well characterize the true continuous operator if the training data is generated by simulations with too coarse spatial resolutions, and thus the prediction results on a higher spatial resolution could lead to larger errors.

The simulation domain is $\Omega = (0,1)^2$ with periodic boundary conditions in both x and y directions, and we simulate the system for the time $t \in (0, L_t]$ with $L_t = 20$. The detailed settings are $\Delta x = 1/256, \Delta y = 1/256, \Delta t = 10^{-4}, \nu = 10^{-3}$, and $f = 0.1(\sin(2\pi(x+y)) + \cos(2\pi(x+y)))$. With initial condition $\omega(x, y, 0)$ randomly sampled from a 2-D Gaussian random field $\mathcal{N}(0, 7^{1.5}(-\Delta + 49I)^{-2.5})$, we perform 1100 simulations in total, with 1000 simulations as training data and the other 100 simulations as test data.

To construct the neural dynamical operator $\tilde{\mathcal{G}}$, we use a 2-D FNO as surrogate model with $k_{max,1} = 12$, $k_{max,2} = 12$, $d_v = 32$ for Resolution 1 to 4 and $k_{max,1} = 8$, $k_{max,2} = 8$, $d_v = 24$ for Resolution 5 in Table. 2.2. The neural dynamical operator is trained with 2×10^4 epochs, and one simulation from the training dataset serves as one mini-batch data (i.e., 1 training batch in each epoch). The optimizer is Adam with 10^{-3} learning rate and cosine annealing schedule.

We train the neural dynamical operator based on time series data with various spatial-temporal resolutions. The test errors are summarized in Table 2.2: the Test Error (I) is based on the test data of the same resolution $\Delta x = 1/64, \Delta y = 1/64$ and $\Delta t = 0.2$, and the Test Error (II) is based on the resolution setting same as each train data. From the Test Error (II), it can be seen that models trained from all resolutions can achieve a small test error and stay at the same order of error magnitude when predicting on a test dataset whose resolution is the same as train data. However, from the Test error (I) in the rows of Resolution 4 and 5, we can see that models trained from a coarse resolution fail to show good performance when testing with higher data resolution. On the other hand, from the Test Error (I) in rows of Resolutions 1 to 3, we can still confirm that the temporal resolution-invariance property is achieved by the trained neural dynamical operator.

Figure 2.6 presents the energy spectrum of data of initial condition from 1-D viscous Burgers' equation and 2-D Navier–Stokes equation with respect to different resolution settings. We can see in Figure 2.6(a) that the energy spectrum of the viscous Burgers' equation is similar for most of the wave numbers with different spatial resolutions. However, Figure 2.6(b) shows more noticeable differences in the energy spectrum of Navier–Stokes equations across the whole range of wave numbers with respect to different spatial resolutions. Unlike spatial discretization of viscous Burgers' equation, where all the resolution settings provide a consistent result of the energy spectrum, the coarse resolution settings of N–S equation can cause over-estimations of energy in low wave numbers, which indicates that the numerical simulations do not well capture the true dynamical operator. Therefore,

Table 2.2: The test errors of Navier–Stokes equation with various resolution settings for train data. The Test Error (I) is based on the test data of the resolution $\Delta x = 1/64$, $\Delta y = 1/64$, and $\Delta t = 0.2$, while the Test Error (II) is based on a resolution setting the same as each training data.

Error / Resolution	Train Data			Test Error (I)		Test Error (II)	
	Δx	Δy	Δt	Absolute	Relative	Absolute	Relative
Resolution1	1/64	1/64	0.2	2.2011e-04	2.6937e-02	2.2011e-04	2.6937e-02
Resolution2	1/64	1/64	0.4	2.1764e-04	2.6753e-02	2.1739e-04	2.6827e-02
Resolution3	1/64	1/64	1	2.0775e-04	2.6083e-02	2.0591e-04	2.6089e-02
Resolution4	1/32	1/32	0.2	1.3277e-01	5.0054e-01	2.2525e-04	2.7820e-02
Resolution5	1/16	1/16	0.2	3.5572e-01	7.9330e-01	3.1081e-04	3.2806e-02

the trained models in the example of viscous Burgers' equation can approximate the true continuous operator and adapt well to the different resolution settings in Table. 2.1, with small test errors when making predictions in a higher spatial resolution. However, in the example of N–S equation, the trained models from coarse resolution settings in Table 2.2 provide large test errors when making predictions in a higher spatial resolution, mainly due to the information loss in high wave numbers that prevent a good approximation of the true continuous operator.

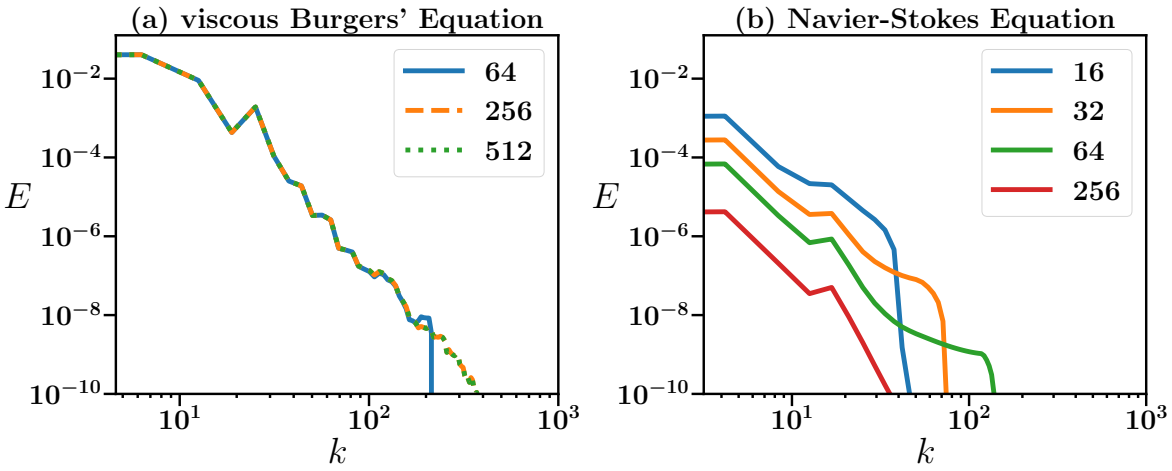

Figure 2.6: Energy spectrum of initial condition data of viscous Burgers' equation and Navier–Stokes equation with respect to different resolution settings in the Table 2.1 and Table 2.2.

In Figure 2.7, we present the spatial-temporal plots of the solutions with a spatial

resolution $\Delta x = 1/16, \Delta y = 1/16$ for true system and the model trained by Resolution 5 in Table 2.2, with the initial condition sampled from test data. The upper row presents the true solution of ω. The lower row corresponds to the prediction from the trained model. We can see that the trained model can capture the overall pattern of the true system in the spatial resolution 16×16. The error between upper and lower in Figure 2.7 corresponds to the Test Error (II) with Resolution 5 in Table. 2.2. It should be noted that the models trained by other resolutions in Table 2.2 also have good prediction results in the same resolution as the corresponding training data, which are omitted here for simplicity.

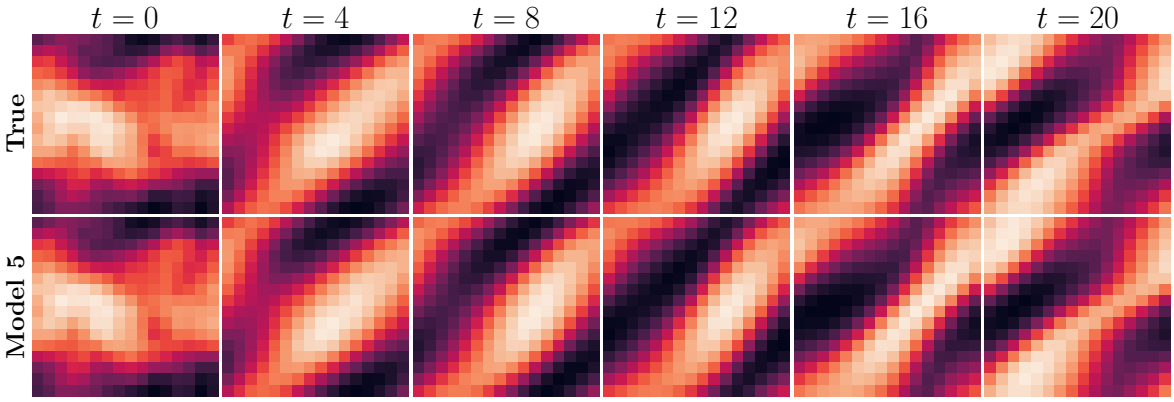

Figure 2.7: The spatial-temporal true simulation and model prediction. Upper row: true system with spatial resolution 16×16. Lower row: predictions made by models trained with the data in a spatial resolution 16×16 and tested on the data in the same resolution.

We then present the spatial-temporal plots with the spatial resolution $\Delta x = 1/64, \Delta y = 1/64$ and the temporal resolution $\Delta t = 0.2$ in Figure 2.8 for the true system and the prediction results of trained models. The initial condition is the same as the one used in Figure 2.7 but with a finer resolution $\Delta x = 1/64, \Delta y = 1/64$. The first row presents the true solution of ω, and the other three rows correspond to the prediction results of the trained models with Resolutions 3, 4 and 5 in Table. 2.1. We can see that only Model 3 in Figure 2.8 can capture the flow pattern of the true system, while Model 5 displays a noticeable mismatch with the true solution. Compared with Figure 2.7, we can see that the results of Model 5 show a good performance for test data in spatial resolution $\Delta x = 1/16, \Delta y = 1/16$ (which is the same as train data), while the prediction results are unsatisfactory for the spatial resolution $\Delta x = 1/64, \Delta y = 1/64$. On the other hand, the good prediction results of Model 3 confirm that the trained model is temporal-invariant, i.e. capable of adapting to data with different temporal resolutions even with relatively sparse spatial data.

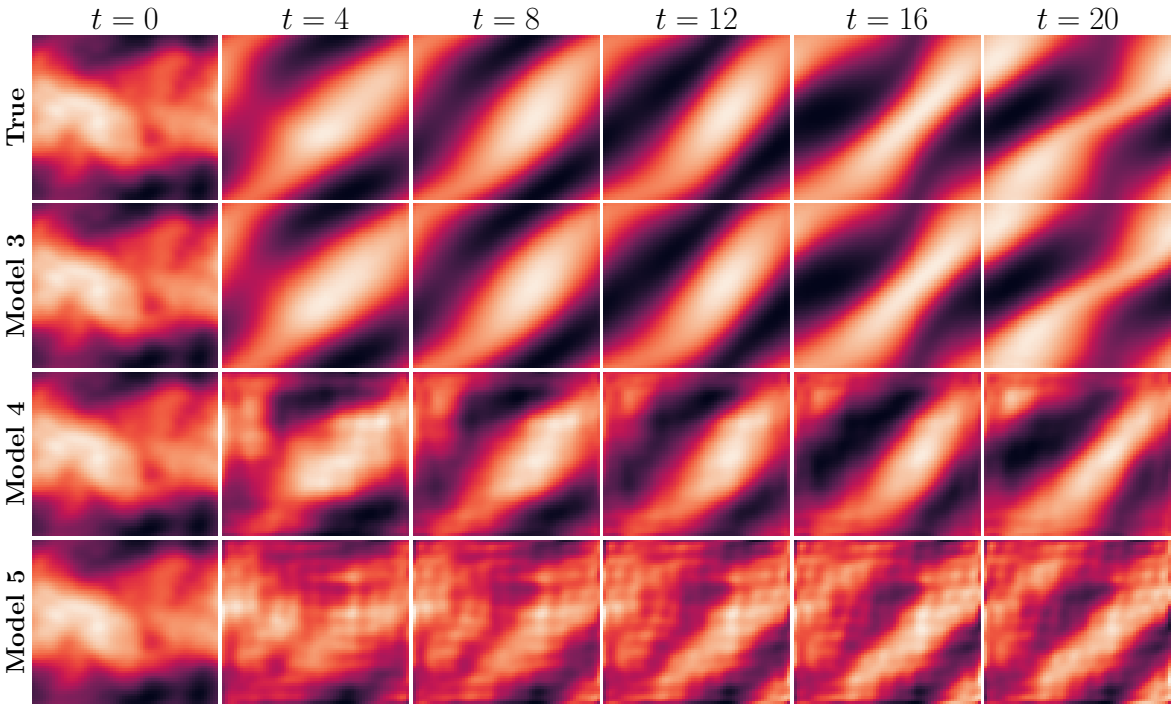

Figure 2.8: The flow of true simulation and model predictions of N-S equation from 0 to 20 time units. The predictions are made by models trained from different resolution settings in Table 2.2.

We further compare the energy spectrum of the solutions between the true system and the trained models in Figure 2.9. A reference slope k^{-3} is also included, which corresponds to the empirical decay rate of 2-D turbulence based on experimental data. The three trained models are tested with the same spatial-temporal resolution (i.e., $\Delta x = 1/64, \Delta y = 1/64, \Delta t = 0.2$). We can see that only the energy spectrum of Model 3 has a good agreement with the true spectrum, while the results of Models 4 and 5 both demonstrate noticeable differences from the true one. It should be noted that the energy spectrum of the prediction results from Models 4 and 5 also do not agree well with the true results in Figure 2.6(b), which indicates that the trained neural dynamical operator only based on data in very coarse resolutions may not generalize well to the finer resolutions.

2.6.3 Kuramoto—Sivashinsky Equation

Kuramoto-–Sivashinsky (K–S) equation [167–170] is a fourth-order nonlinear partial differential equation that was originally developed to model diffusive–thermal instabilities in a

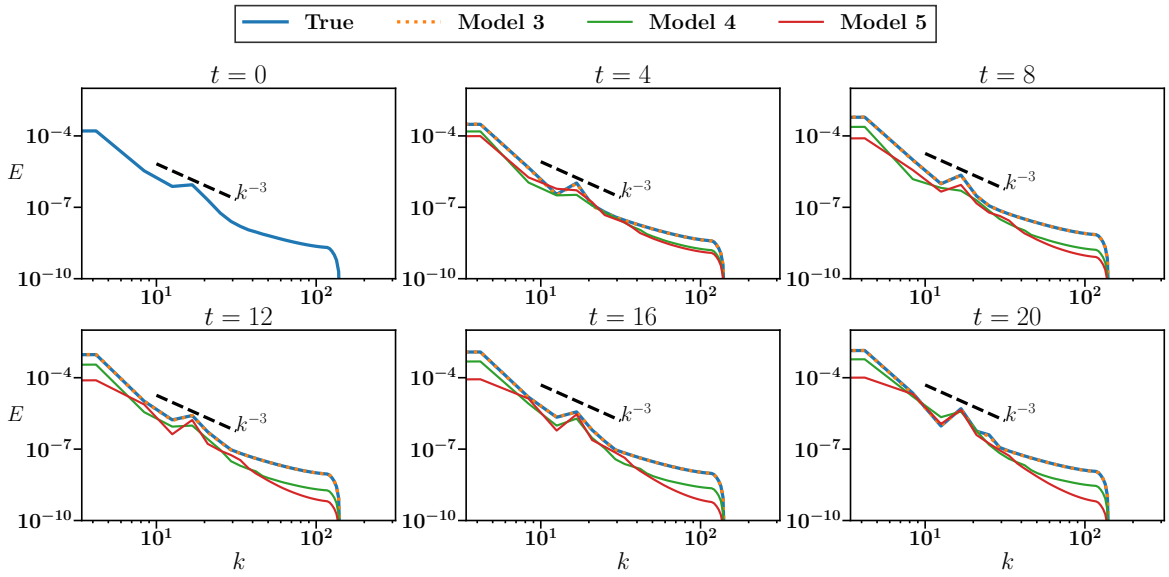

Figure 2.9: Energy spectrum of 2-D Navier–Stokes equation simulated with an initial condition from test data at various times.

.

laminar flame front and features chaotic behavior and rich dynamics, e.g., dissipation and dispersion. The governing equation of the K-S equation is:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}, \tag{2.17}$$

where $x \in (0, L_x)$ with periodic boundary conditions, $t \in (0, L_t]$ and $u(x, 0)$ is the given initial condition. We aim to learn a neural dynamical operator to approximate the right-hand-side of (2.17), i.e., $\mathcal{G} : u \mapsto \frac{\partial u}{\partial t}$. In this example, we demonstrate that (i) the trained neural dynamical operator is spatial-temporal resolution-invariant and can provide good short-term prediction, (ii) the trained model based on short-term data can provide a stable long-term simulation with the chaotic behavior being retained qualitatively, and (iii) the model can achieve good performance for both short-term trajectory prediction and long-term statistics matching when trained with a hybrid optimization method. These three features are described in detail in the subsequent sections.

The simulation settings are $L_x = 22, L_t = 5000, \Delta x = 22/1024, \Delta t = 0.025$ and the initial condition is $u(x, 0) = 0.1 \times \cos(x/16) \times (1 + 2\sin(x/16))$. We simulate the true system with a single long trajectory, and the first 80% of the trajectory (4000 time units) is used as train

data and the remaining 20% (1000 time units) is used as test data. For the K-S equation, 1000 time units is long enough to demonstrate the stability of long-term prediction of neural dynamical operator.

2.6.3.1 Short-Term Prediction with Spatial-Temporal Resolution-Invariant

We first train the model for short-term state prediction by solving the optimization problem in (2.3). Two short-term sub-trajectories of 20 time units will be sampled from train time series data to serve as one data batch. The neural dynamical operator $\tilde{\mathcal{G}}$ is constructed by a FNO model with $d_v = 64$ and $k_{max} = 24$. We train the model with 2×10^4 epochs, and the optimizer is Adam with a learning rate 10^{-3} and cosine annealing schedule.

We train the neural dynamical operator based on short-term time series data with various resolutions in both space and time. The test results are summarized in Table 2.3. The absolute test error is the mean squared error between true and predicted values for 20 time units in test data. The long-term D_{KL} is the Kullback–Leibler (KL) divergence (defined in (2.18)), which quantifies how a probability distribution differs from the other. Given PDF $p(x)$ from true data and $\tilde{p}(x)$ from predicted data, the formula for the forward KL divergence of distributions $\tilde{p}(x)$ from $p(x)$ is:

$$D_{KL}(p||\tilde{p}) = \int_{-\infty}^{\infty} p(x) \log(\frac{p(x)}{\tilde{p}(x)}) dx, \tag{2.18}$$

which is estimated from samples of both distributions based on k-Nearest-Neighbours probability density estimation [171]. The forward KL divergence in (2.18) is mean-seeking, while the reverse KL divergence $D_{KL}(\tilde{p}||p)$ is mode-seeking. In this work, both forward and reverse KL divergences are estimated based on 1000 time units in test data. Each point in Figure 2.14 stands for D_{KL} from PDF of true data and PDF of predicted data.

We summarize the short-term test errors and long-term forward KL divergence for the system state u in Table 2.3. The Test Error (I) and Long-Term D_{KL} (I) are based on the test data of same resolution $\Delta x = 1/1024$ and $\Delta t = 0.05$, and the Test Error (II) and Long-Term D_{KL} (II) is based on a resolution setting the same as the one of each train data. By comparing these test errors for trained models on different resolutions, it can be seen that the test error stays at the same order of magnitude when testing on a finer resolution in both space and time, confirming the resolution-invariance property of the trained models. In addition, all trained models lead to stable long-term simulations, which is mainly due to the high

wavenumber filtering at each time step of evaluating the neural dynamical operator. More importantly, the stable long-term simulations of the trained models demonstrate small errors in KL divergence of the system state u, indicating a good quantitative agreement for the long-term prediction of the system state with the initial conditions from the test data. The visualization of D_{KL} (I) is shown in the first plot of Figure 2.12. It can be observed that the probability density functions (PDFs) obtained from models trained with the three different resolutions are very similar.

Table 2.3: The test errors of Kuramoto—Sivashinsky equation with various resolution settings for train data. The Test Error (I) and Long-Term D_{KL} (I) are based on the test data of the resolution $\Delta x = 22/1024$ and $\Delta t = 0.25$, while the Test Error (II) and Long-Term D_{KL} (II) are based on a resolution setting the same as each training data.

Error / Resolution	Train Data		Test Error (I)		Long-Term D_{KL} (I)	Test Error (II)		Long-Term D_{KL} (II)
	Δx	Δt	Absolute	Relative		Absolute	Relative	
Resolution1	22/1024	0.5	4.6195e-02	1.3609e-01	1.6776e-03	4.5221e-02	1.3442e-01	1.6064e-03
Resolution2	22/512	1	6.3813e-02	1.4419e-01	5.1330e-03	5.9550e-02	1.3860e-01	1.5060e-03
Resolution3	22/256	2	5.6369e-02	1.2588e-01	8.6506e-03	4.9447e-02	1.1480e-01	1.7834e-03

The Long-Term D_{KL} (I) and (II) listed in the Table 2.3 are the forward KL divergence. The corresponding reverse KL divergence for each resolution have also been calculated. The reverse Long-Term D_{KL} (I) are 3.5586e-03, 6.2831e-03, 8.4653e-03 and reverse Long-Term D_{KL} (II) are 3.3100e-03, 4.9920e-03, 1.6984e-03.

To facilitate a more detailed comparison between the short-term simulations of the true system and the modeled ones, we present the solution profiles u in Figure 2.10 for 20 time units with three initial conditions (i.e., $t_0 = 4000, 4500, 4900$) from the test data. Each row corresponds to a different test initial condition, and each model corresponds to a resolution setting with the same index in Table 2.3. The trained models are tested with a finer resolution $\Delta x = 22/1024$ and $\Delta t = 0.25$. We can see that the solution profiles of the trained model at various times all have a good agreement with the true solution profiles, with only some small deviation at a few regions. The result in Figure 2.10 confirm that the trained neural dynamical operator has resolution-invariance in both space and time and also generalizes well to initial conditions from the test data.

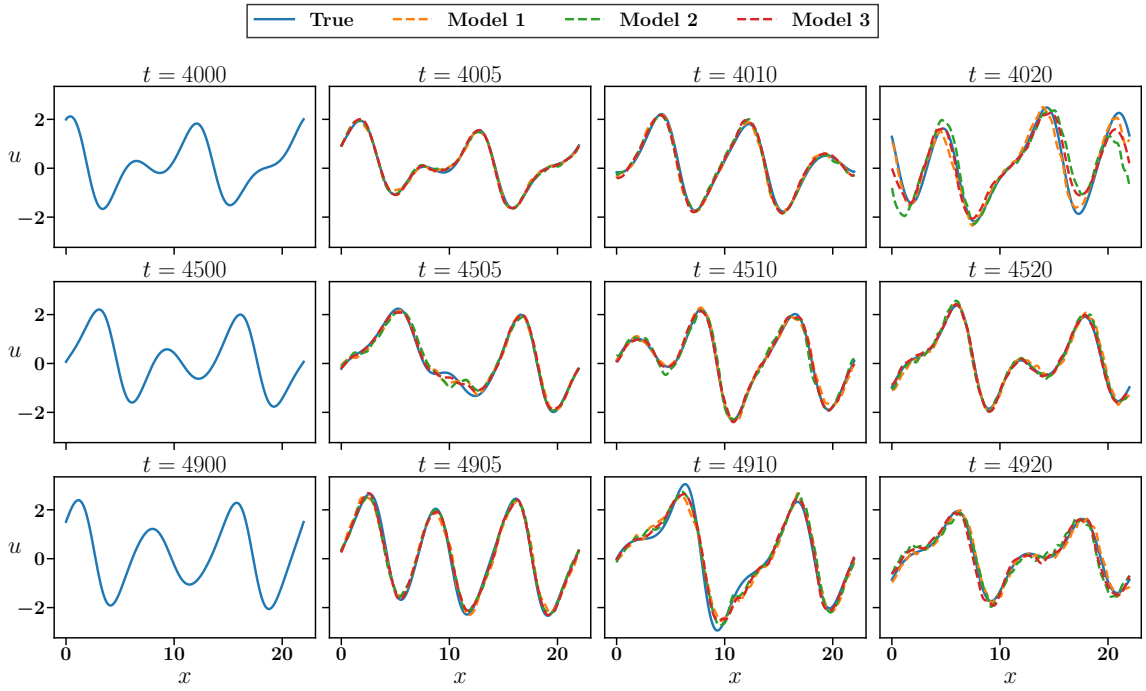

Figure 2.10: Solution profiles of the K-S equation for the true system and the model ones with different initial conditions from test data. The model is trained with each resolutions in Table. 2.3 and tested on a finer resolution ($\Delta x = 22/1024, \Delta t = 0.25$).

2.6.3.2 Stable Long-Term Simulation with Chaotic Behavior

In Figure 2.11, we present 500 time units of spatial-temporal solutions from the true system and the modeled ones trained by the short-term time series data (20 time units), with the initial condition sampled from the test data. Considering that the K-S equation is a chaotic system, we would not expect a good quantitative agreement of long-term trajectories between the true system and the modeled one. It can be seen in Figure 2.11 that the patterns of 500 time units spatial-temporal solution plots demonstrate a good qualitative agreement with the true system, even though the models are trained with much shorter trajectories of the system state.

We further examine the statistical properties of long-term solutions from true and model systems in Figure 2.12. Based on the true simulations and predictions for 1000 time units, we compare the probability density function (PDF) of the system state u, first-order spatial derivative u_x, second-order spatial derivative u_{xx}, and spatial and temporal auto-correlation function (ACF) of u. We can find that the PDF of state u and its first spatial derivative from

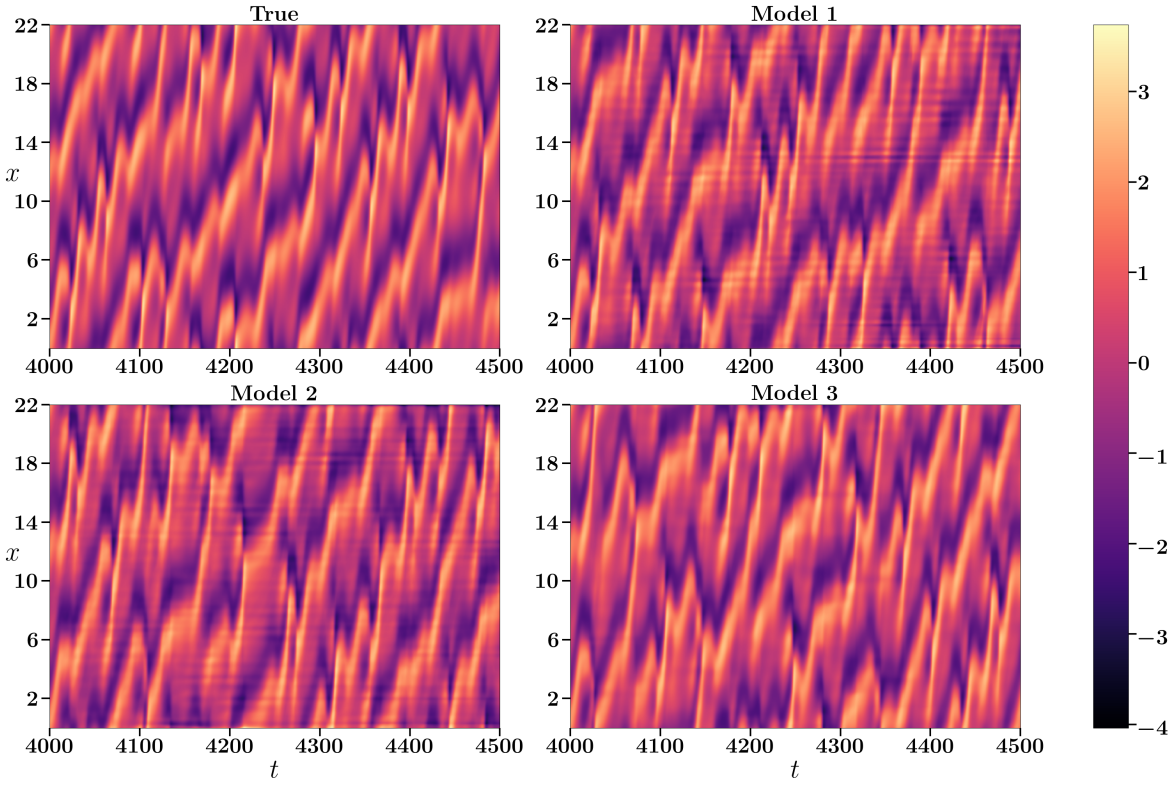

Figure 2.11: The 500 time units spatial-temporal solutions of Kuramoto—Sivashinsky equation. True: the solution simulated from the true system. Model: trained models from three different resolutions with the same test data resolution ($\Delta x = 22/1024, \Delta t = 0.25$). The index in the three trained models corresponds to the resolution settings in Table 2.3.

long-term prediction can match the true simulation well, while the PDF of the second spatial derivative u_{xx} from the modeled systems shows a less satisfactory agreement with the true one. Also, the temporal and spatial ACF of state u show a similar pattern between predicted and true values, indicating a stable long-term prediction by trained models with similar statistical properties. In Figure 2.13, we also show the joint probability density function of (u_x, u_{xx}) for the long-term (1000 time units) simulation from true system and modeled systems with the same resolution $\Delta x = 22/1024, \Delta t = 0.25$. Compared with the joint PDF from true simulation, even though the joint PDFs from model predictions have a relatively lower maximum density and spread more out, their overall patterns are still qualitatively similar to the true system.

Besides the qualitative visualization of those probability density functions, the KL divergence from PDFs of true data to PDFs of model predictions are calculated and summarized

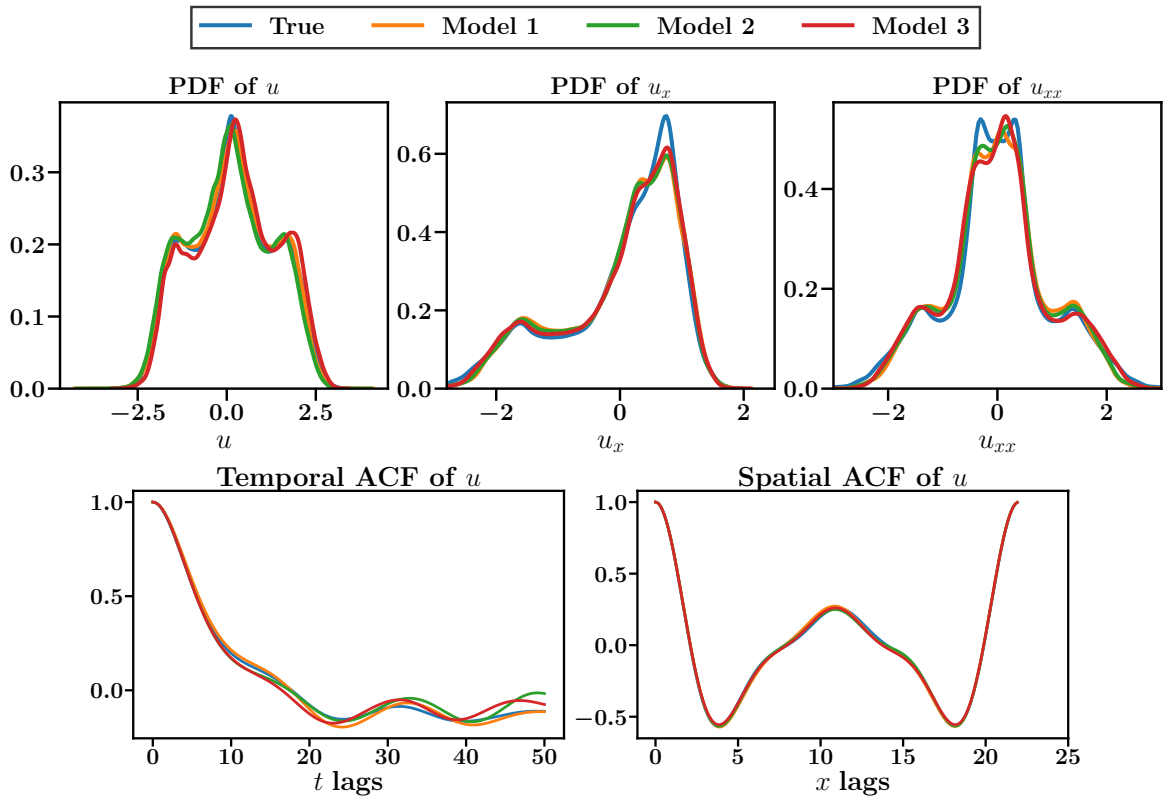

Figure 2.12: Probability density function and auto-correlation function of long-term (1000 time units) true simulation and model predictions for Kuramoto—Sivashinsky equation in test data. Upper: probability density function of state u, first spatial derivative u_x and second spatial derivative u_{xx}. Below: temporal and spatial auto-correlation function of state u.

in Figure 2.14. The left panel displays the forward KL divergence, while the right panel shows the reverse KL divergence. The KL divergence results include u, u_x, u_{xx} and (u_x, u_{xx}) and show small values for all the trained models with different training resolutions, demonstrating the resolution-invariance property of trained neural dynamical operator even in long-term predictions.

2.6.3.3 Hybrid Optimization for Short-Term State Prediction and Long-Term Statistics Matching

The previous two sections show that the neural dynamical operator, trained with abundant short-term data, can make accurate short-term state predictions and consistent long-term

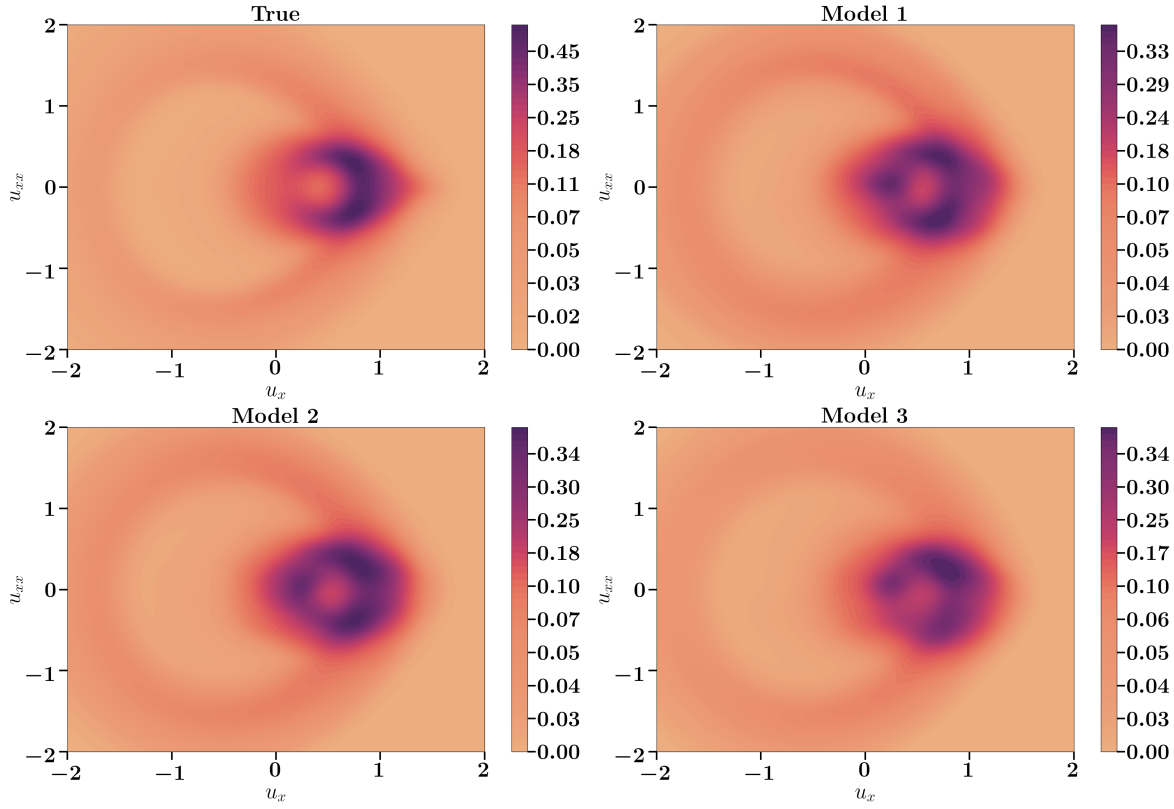

Figure 2.13: Joint probability density function of first spatial derivative and second spatial derivative (u_x, u_{xx}) for long-term (1000 time units) simulations from true and modeled systems in test data. The model systems are trained by resolution settings in Table. 2.3 and tested on the resolution $\Delta x = 22/1024, \Delta x = 0.25$.

statistics with the feature of resolution-invariant if sufficient time series data (4000 time units) is available. To demonstrate the merits of the hybrid optimization, we consider a practical application with data scarcity: only some short-term batches and long-term statistics from the 4000 time units training data in resolution 3 ($\Delta x = 22/256, \Delta t = 2$) are available. More specifically, the data available for use include 40 short-term batches, each with a length of 2 time units, and 20 sets of long-term statistics (variance) from 200 time units with known initial conditions distributed evenly through training data. Trained with only the short-term batches, the neural dynamical operator can still produce relatively accurate short-term state prediction and stable long-term simulation. However, the long-term variance of u_{xx} is much larger than the truth. With the data of long-term variance of u_{xx}, hybrid optimization can effectively calibrate the pre-trained neural dynamical operator

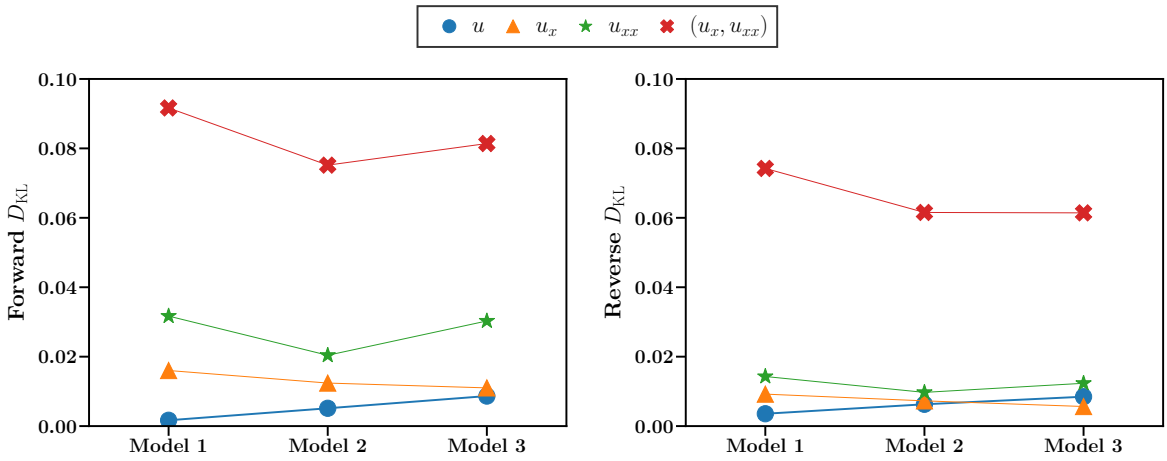

Figure 2.14: Summary of KL Divergence between PDFs from 1000 time units simulation from true system and modeled system. Those PDFs and joint PDF includes u, u_x, u_{xx}, and (u_x, u_{xx}).

to enhance the consistency with the long-term statistics of the true system. The forward map in (2.10) of EKI is constructed by the composition of three components: (i) generating long-term simulations of 200 time units with neural dynamical operator, (ii) calculating the second-order spatial derivative u_{xx} from the simulated system state u, and (iii) calculating the variance of u_{xx}.

Starting with the pre-trained model with short-term loss only, the hybrid optimization updates the neural dynamical operator by alternating the short-term trajectory matching via gradient-based optimization (i.e., Adam optimizer in this work) and the long-term statistics matching via derivative-free optimization (i.e., EKI in this work). More details about the hybrid optimization algorithm have been summarized in Algorithm 3. The number of total training epochs is $N_{\mathrm{Hybrid}} = 10$ with $N_{\mathrm{SGD}} = 300$ of gradient descent updates and then $N_{\mathrm{it}} = 20$ EKI updates in each epoch. The learning rate of the Adam optimizer is set as 10^{-4}, considering that we are tuning a well-trained model at the beginning, and the ensemble size of EKI is $J = 100$.

We present the error history during the EKI updating in Figure 2.15. The short-term error is the mean squared error for short-term (2 time units) system state trajectory, while the long-term error is the mean squared error of variances of u_{xx} from long-term simulations with 200 time units. We can see that there is a trade-off between short-term state error and long-term statistics error in each EKI epoch, because each EKI epoch only focuses

on long-term statistics matching in the proposed hybrid optimization method. Although the short-term error tends to increase within each EKI epoch, the subsequent epochs of gradient-based optimization with short-term trajectory matching would keep tuning the neural dynamical operator such that a smaller short-term error is achieved. With the EKI loss training history, we selected the parameters trained after two iterations in 10^{th} EKI epoch, highlighted by the red circle in the third column of Figure 2.15. It is worth noting that the proposed method sequentially optimizes the short-term loss L_s and the long-term loss L_l individually and then chooses a robust result, i.e., relatively small values of L_s and L_l compared to their range of values, respectively. In general, the approach is related to solving a two-objective optimization problem based on the Pareto front, which is estimated via iteratively solving the optimization problem that only involves L_s or L_l.

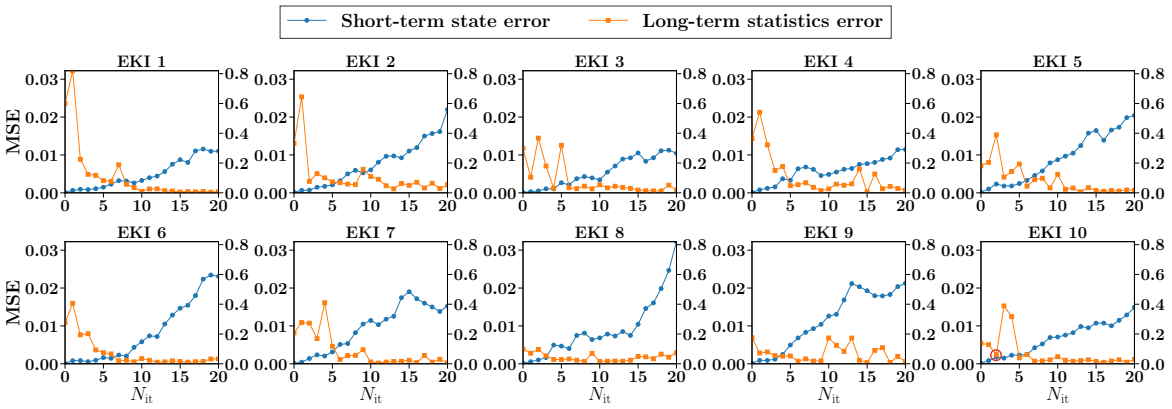

Figure 2.15: Long-term and short-term error history of the EKI epochs in the hybrid optimization. In each EKI epoch, the parameters will be updated 20 iterations based on (2.12). The 0-th iteration is the error of the model updated by the previous gradient-based optimization epochs. In each sub-figure, the right y-axis is the short-term state MSE, and left y-axis is the long-term statistics MSE.

The relationship between long-term statistics error and short-term state error during the EKI training is presented in Figure 2.16. Each sub-figure corresponds to its counterpart in Figure 2.15. In each EKI iteration, we observe a decreasing trend of long-term statistics error while an increase in short-term error. This again manifests the trade-off between long-term and short-term errors during the EKI updating process. In this section, we select the trained model that provides a balanced performance of short-term trajectory and long-term statistics, which corresponds to the red point of the 10^{th} EKI epoch at the bottom-left region

(i.e., relatively small short-term and long-term errors) in Figure 2.16. The selected model is also highlighted by the red circle in Figure 2.15.

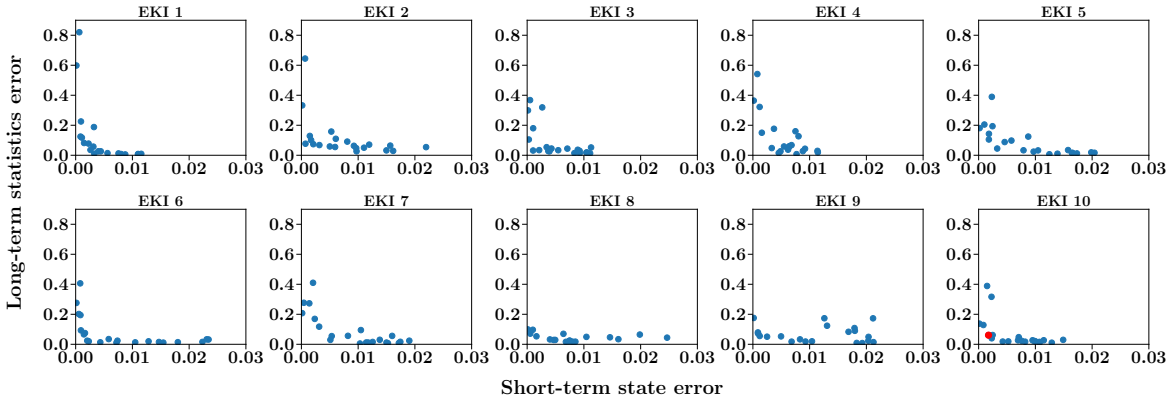

Figure 2.16: Long-term statistics error versus short-term state error during the EKI updating in the hybrid optimization. Each sub-figure corresponds to the counterparts in the Figure 2.15.

The short-term solution profiles from the simulation of the true system and predictions of models with classical and hybrid optimization schemes are presented in Figure 2.17. The simulation of the true system has the resolution $\Delta x = 22/256, \Delta t = 2$, and both models are trained with the the 40 short-term batches with 2-time-unit length from the simulation. The three initial conditions are from test data the same as Figure 2.10. From the comparison of solution profiles starting with those initial conditions in Figure 2.17, we can find that short-term predictions from both models are similar to the true solution profiles when the predictive horizon less than 20 time units. The model trained solely with classical optimization performs slightly better than the model with hybrid optimization. More specifically, the absolute and relative short-term state prediction errors for 20-time-units of the model with classical optimization are 0.2765 and 0.2865, while the errors of the model with hybrid optimization are 0.3601 and 0.3565, respectively.

With a comparable performance of short-term prediction to the model with classical optimization, hybrid optimization can lead to better long-term statistics as presented in Figure 2.18. More specifically, the mean squared error of the long-term statistics (i.e., the variance of u_{xx}) predicted by the model with classical optimization is 0.5340, while the one with hybrid optimization is 0.0187. This indicates that the model with hybrid optimization has a much better agreement with the true value of the variance than the one with solely

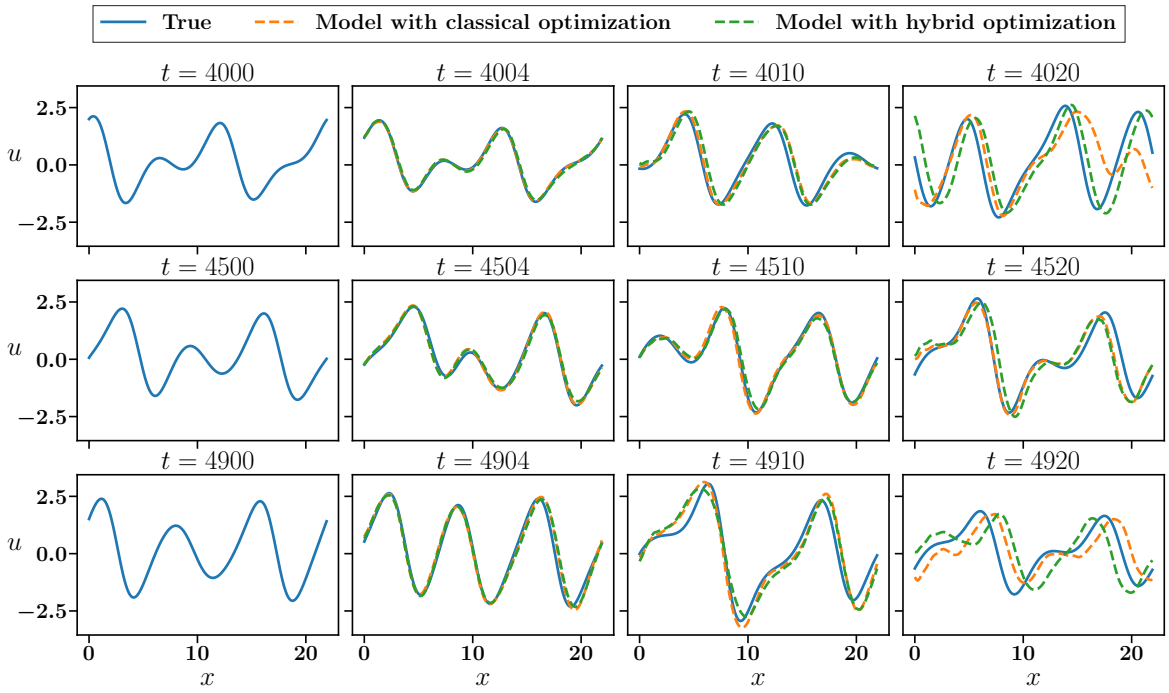

Figure 2.17: Solution profiles of the K-S equation for the true system, model trained with classical optimization, and model trained via hybrid optimization with different initial conditions in test data. The model with classical optimization is trained with 40 short-term batches with length of 2 time units from training data with resolution $\Delta x = 22/256, \Delta t = 2$. The model with hybrid optimization is further calibrated using the 20 sets of long-term statistics data. Both models are tested on the same resolution.

classical optimization. We present the probability density distribution of the second spatial derivative u_{xx} for 1000 time units test data from the true system and the results of the two trained models in Figure 2.18. It should be noted that the PDFs from models are based on five sets of 200-time-unit simulations with initial conditions distributed evenly through the test data. The peak part of the PDF from the model with hybrid optimization is higher than the model with classical optimization, and its tail part is also more in line with the true PDF. This improvement contributed by the hybrid optimization approach indicates that the trained model is more capable of quantifying the dispersion and uncertainty of the true system, e.g., extreme events that are of interest in science (e.g., extreme weather/climate) and engineering (e.g., responses of materials or energy systems with extreme loads) applications.

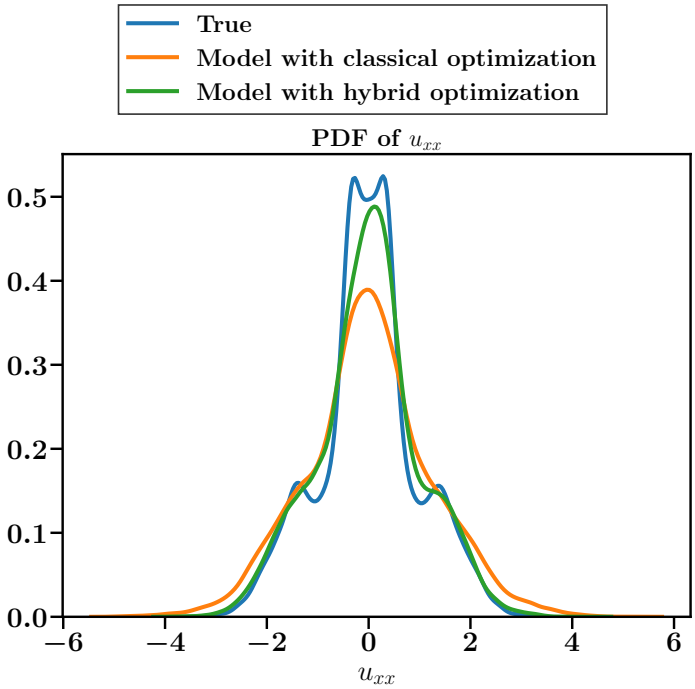

Figure 2.18: Probability density function of second spatial derivative u_{xx} from long-term (1000 time units) simulation of true system, five sets of 200 time units simulation from model trained with classical optimization and model trained with hybrid optimization with initial conditions distributed evenly from test data.

2.7 Conclusion

A recent trend of data-driven modeling is to formulate the problem in its continuous form, which facilitates a more flexible use of data in general. The merits of existing spatially continuous models (e.g., neural operator) and temporally continuous models (e.g., neural ODE) have been demonstrated in many science and engineering applications. In this work, we present a data-driven modeling framework that learns a continuous spatial-temporal model based on the techniques of neural operator and neural ODE. More specifically, we focus on the learning of the dynamical operator and demonstrate that the learned model is resolution-invariance in both space and time. We also show that the learned model can provide stable long-term simulations, even if the training data only contains short-term time series of true system states. In addition, we propose a hybrid optimization scheme that leverages both gradient-based and derivative-free methods and efficiently combines the use of short-term time series and long-term statistics in training the model.

The proposed framework is studied based on three classical examples governed by partial differential equations, including the viscous Burgers' equation, the Navier–Stokes equations, and the Kuramoto–Sivashinsky equation. The results show that: (i) the trained model has resolution-invariance with respect to both spatial and temporal discretizations, and (ii) the hybrid optimization scheme ensures a good performance of the trained model in both matching short-term trajectories and capturing long-term system behaviors. Although this work mainly focuses on demonstrating the use of FNO to construct the neural dynamical operator and the iterative optimization of short-term and long-term losses, it is worth noting that other operator learning techniques (e.g., DeepONet) and optimization methods (e.g., ϵ—constrained method for multiobjective optimization) can also be employed by the proposed framework.

Chapter 3

Online Sparse Identification of Regime-Switching Dynamical Systems via a Causal Approach

Regime switching is ubiquitous in many complex dynamical systems with multiscale features, chaotic behavior, and extreme events. In this paper, a causation entropy boosting (CEBoosting) strategy is developed to facilitate the detection of regime switching and the discovery of the dynamics associated with the new regime via online model identification. The causation entropy, which can be efficiently calculated, provides a logic value of each candidate function in a pre-determined library. The reversal of one or a few such causation entropy indicators associated with the model calibrated for the current regime implies the detection of regime switching. Despite the short length of each batch formed by the sequential data, the accumulated value of causation entropy corresponding to a sequence of data batches leads to a robust indicator. With the detected rectification of the model structure, the subsequent parameter estimation becomes a quadratic optimization problem, which is solved using closed analytic formulae. Using the Lorenz 96 model, it is shown that the causation entropy indicator can be efficiently calculated, and the method applies to moderately large dimensional systems. The CEBoosting algorithm is also adaptive to the situation with partial observations. It is shown via a stochastic parameterized model that the CEBoosting strategy can be combined with data assimilation to identify regime switching triggered by the unobserved latent processes. In addition, the CEBoosting method is applied

to a nonlinear paradigm model for topographic mean flow interaction, demonstrating the online detection of regime switching in the presence of strong intermittency and extreme events.

3.1 Introduction

Regime switching is ubiquitous in many complex dynamical systems in geoscience, engineering, neural science, and material science [3, 172–177]. The switching is usually associated with sudden changes in internal states or appears when specific external forcing is exerted. Dynamical systems often display distinct behavior with regime switching. One example is the atmospheric jets, which meander in different directions when the atmosphere alternates between blocked and unblocked regimes [178, 179]. Similarly, an excitable medium is susceptible to finite perturbations, which triggers regime switching from a quiescent state to one with various wave patterns [180–182]. Regime switching can also induce an increased occurrence of extreme events, leading to, for example, extreme weather and climate patterns [183, 184], bursting neurons [185], or extreme ductile damages [186]. Detecting regime switching and the corresponding underlying dynamics, which relies on appropriate model identification methods, has significant social and scientific impacts. Challenges in detecting regime switching are associated with the intrinsic properties of many complex dynamical systems, including high dimensionality, partial or incomplete observations, and the intermittent occurrence of rare and extreme events [2, 11, 65, 187, 188].

Efficient model identification has received significant attention. Both physical knowledge and observational data facilitate learning the underlying model dynamics. The model structures are often established utilizing physical intuitions for traditional knowledge-based model identification. The primary process then becomes the estimation of model parameters. Linear models are natural candidates for simple problems [189, 190] and can potentially be skillful for short-term forecasts. Other families of models with pre-determined structures, such as the physics-constrained nonlinear regression models [191, 192] and conditional Gaussian nonlinear systems [193, 194], are alternative nonlinear models aiming to capture specific underlying dynamical features. On the other hand, recent progress has been made in data-driven model identification. Data-driven reduced-order models have been widely used in scientific and engineering applications [77–79, 195]. Sparse model discovery methods also appear as advanced model identification tools that allow automatic learning of the

model structure and parameters from data and lead to nonlinear models with parsimonious structures via sparse regression [15, 16, 105, 196–200]. Recently, an information-flow-based approach for system identification has gained popularity which utilizes causation entropy to identify the sparse model structure of an unknown dynamical system [201]. An iterative approach called entropic regression which is robust with noise and outliers has further been developed and applied to the area of cognitive neuroscience to improve the accuracy of recovery of the structure of structural and functional connectivity between anatomical regions [202–204]. With a limited amount of indirect data, derivative-free optimization methods [67, 205, 206] have been explored as model identification tools. In addition, non-parametric and machine learning models have been built to characterize complex dynamical systems [32, 33, 207–210].

Among various model discovery approaches, online model identification is a particularly useful method in practice, which sequentially determines model structure and estimates model parameters when new observation arrives [211–216]. It is the primary model identification strategy in many geophysical and engineering problems, where the limited amount of historical data is insufficient to robustly discover the underlying dynamics. It should be noted that online model identification can be further combined with data assimilation to handle noisy observations or recover the unobserved state variables in the situation with partial observations [66, 217, 218]. However, unlike the online parameter estimation that can be efficiently addressed by standard filtering methods [62], the lack of knowledge about the proper model structure poses a unique challenge in online model identification. The sequentially arriving data plays a vital role in progressively rectifying the model and reducing the uncertainty in the identified system. Although existing system identification methods can identify the proper model structure via promoting sparsity in the offline setting with fitting a model to abundant data, promoting sparsity relies on the model fitting may eliminate some important model structures in the context of sequential learning. To address such a challenge, we incorporate causation entropy to achieve robust online model identification. As regime switching often occurs and completes within a short transient period, developing suitable online identification methods for discovering regime switching exploiting transition data is essential with practical importance.

In this paper, a causation entropy boosting (CEBoosting) strategy is developed. It is incorporated into an online model identification method to detect regime switching and discover the nonlinear dynamics associated with the new regime. Different from many

existing sparse model identification algorithms, such as those relying on LASSO (least absolute shrinkage and selection operator) regression [67, 219] or thresholding [16, 105], the method developed here separates the estimation of model parameters from the recurrent identification of nonlinear model structure. Such a separation allows using closed analytic formulae for the entire online learning algorithm, and therefore, the overall computational cost is significantly reduced. In this new strategy, causation entropy [220, 221] is utilized to provide a logic value (i.e., true or false) of each candidate function in a pre-determined library throughout the online learning process. By examining the causation entropy on the newly arrived data, the reversal of one or a few such causation entropy indicators associated with the model calibrated for the current regime implies the detection of regime switching. In other words, the causation entropy indicator, which can be efficiently calculated, is employed to decide if the existing terms in the current model need to be rectified and if the system demands additional terms as a response to regime switching. Note that the sequential data in online learning is collected within a short time window to form a batch of time series, which is utilized to compute the causation entropy. As each batch contains a short amount of data, it may embody only part of the dynamical properties. Nevertheless, as time evolves, the accumulated value of causation entropy corresponding to a sequence of batches leads to a robust indicator of the model structure in response to regime switching. The concept of accumulating causation entropy calculated from sequential data relates to the statistical method of bagging [222]. With the detected rectification of the model structure, the subsequent parameter estimation becomes a quadratic optimization problem, which is solved using closed analytic formulae. For multiple times of regime switching, a summation of residual models is calibrated, which relates to the statistical method of boosting [223–226].

The proposed new strategy has several unique features. First, causation entropy takes into account the interdependence between all the candidate functions in the pre-determined library, and therefore, it can eliminate the superficial causal relationship. Model identification exploiting the causation entropy has been shown to reach a higher selection accuracy than LASSO regression or elastic net [221]. The causation-based learning approach also indicates robust results in the presence of indirect coupling between features and stochastic noise [227], which are crucial features of complex systems. Second, causation entropy is only utilized to indicate the terms that need to be added or removed from the existing model. In other words, although computing the exact value of the causation entropy is challenging, closed analytic

formulae are available for efficiently approximating this causation entropy indicator, which allows an effective detection of the model structure. Third, the parameter estimation only needs to be carried out after the model structure is entirely determined. Therefore, the overall computational cost is reduced compared with applying LASSO regression, which requires detecting the model structure and estimating model parameters simultaneously for each batch of data. Lastly, compared to the pioneering works of using causation entropy for system identification (e.g., [20, 201, 220]), this work focuses on dynamical systems with regime switching, and the proposed CEBoosting method utilizes online data to (i) identify the regime switching and (ii) estimate the sparse model structure to calibrate the unknown model parameters for the new regime. Therefore, the key contribution of this work is to demonstrate the use of causation entropy for a robust online sparse system identification with the sequential and limited amount of data, by leveraging statistical concepts of bagging and boosting. It is worth highlighting that the causation entropy indicator is easy to calculate and applicable to moderately large dimensional systems. The method developed here is also adaptive to the situation with partial observations, where utilizing data assimilation to recover the unobserved state variables can be incorporated into the learning process for identifying regime switching resulting from the latent processes. In addition, the method is not limited to the Gaussian data. It can be applied to dynamical regimes with strong intermittency and extreme events. Applications to nonlinear dynamical systems with moderately large dimensions, partial observations, and extreme events are all studied in the paper.

3.2 Problem Statement

The online sparse identification method aims to (i) detect regime switching of dynamical systems via causation entropy and (ii) determine the resulting dynamics after the regime switching. The dynamical system has the following general form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \boldsymbol{\sigma}\dot{\mathbf{W}}(t), \tag{3.1}$$

where $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_p(t)]^\top \in \mathbb{R}^p$ is the multi-dimensional state variable and $\dot{\mathbf{x}}(t) \in \mathbb{R}^p$ is the associated temporal derivative, $\mathbf{f} : \mathbb{R}^p \mapsto \mathbb{R}^p$ is a vector-valued nonlinear function (i.e., vector field) of the state variable, $\dot{\mathbf{W}}(t) \in \mathbb{R}^q$ is a white noise vector and

$\boldsymbol{\sigma} \in \mathbb{R}^{p\times q}$ is a matrix of noise magnitudes. In the absence of random noise forcing, $\boldsymbol{\sigma}$ becomes a zero matrix. Assume that the regime switching occurs at $t = t_s$, i.e., the vector field $\mathbf{f}$ of the original dynamical system changes to $\mathbf{f}^*$. The goal of this work is to detect such a regime switching and to identify the new model after the regime switching.

As regime switching often results from a sudden change of a small number of the model parameters or specific components of the model structure, the residual model $\delta\mathbf{f} = \mathbf{f}^* - \mathbf{f}$ typically has a sparse structure that can be calibrated using relatively short data, which is precisely the case of the online identification problems. Therefore, instead of learning the entire model associated with the new regime, the focus is to estimate the residual part $\delta\mathbf{f} = \mathbf{f}^* - \mathbf{f}$. Once the residual part $\delta\mathbf{f}$ is identified, it is then added to the existing model that provides the new system as a response to the regime switching.

The limited amount of data is assumed to arrive sequentially in the form of batches. The k-th batch represents data $\mathbf{x}(t)$ (or a subset of the vector $\mathbf{x}(t)$ in the partial observation case) for $t \in [t_{B_k}, t_{B_{k+1}})$. It should be noted that, as t_s is typically unknown in practice, the identification algorithm usually does not start from t_s (namely the left point of the first interval $t_{B_1} \neq t_s$). Yet, for the simplicity of presentation, t_{B_1} is chosen to be t_s for the numerical examples in this work. This will not affect the identification algorithm as applying the algorithm to those batches prior to t_s will not indicate regime switching. But this setup facilitates counting for the length of the data that is needed to detect the regime switching once it occurs at t_s.

Denote by $\boldsymbol{\Phi} = [\phi_1, \phi_2, ..., \phi_N]^\top$ a vector containing all candidate basis functions, which are knowledge-based and are pre-determined. Each ϕ_n in $\boldsymbol{\Phi}$ is a scalar-valued function $\phi_n := \phi_n(\mathbf{x})$ that gives a map $\mathbb{R}^p \mapsto \mathbb{R}$. The representation of $\delta\mathbf{f} = \mathbf{f}^* - \mathbf{f}$ is approximated by a linear combination of these basis functions:

$$\delta f_i = \sum_{n=1}^{N} \delta\xi_{in}\phi_n, \tag{3.2}$$

where δf_i is the i-th scalar component of $\delta\mathbf{f}$. A sparse representation of (3.2) means that most of the coefficients $\delta\xi_{in}$ are zeros in the identified model. To obtain such a sparse representation of (3.2), a CEBoosting method is developed to effectively determine which basis functions should take non-zero coefficients.

3.3 Causation Entropy for System Identification

Causation entropy is based on the general concept of conditional mutual information [228]. Other popular information measures that also build on conditional mutual information include directed information [229] and transfer entropy [230], which has found applications in many areas, e.g., turbulence modeling [231, 232] and neurosciences [233], with a comprehensive review in [234]. The idea of utilizing the causation entropy to detect the influence between different variables has been developed in [201–204] and studied in [20, 220, 221, 235–237]. It can be naturally applied to the context of system identification. As the white noise does not explicitly contribute to the causal relationship, the calculation of the causation entropy mainly focuses on the candidate functions that consist of the deterministic part of the dynamics, namely the function $\mathbf{f}$ in (3.1). To this end, consider the deterministic part of (3.1):

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \end{bmatrix} = \begin{bmatrix} \xi_{1,1} & \cdots & \xi_{1,N} \\ \xi_{2,1} & \cdots & \xi_{2,N} \\ \vdots & \ddots & \vdots \\ \xi_{p,1} & \cdots & \xi_{p,N} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix} = \boldsymbol{\Xi}\boldsymbol{\Phi}. \tag{3.3}$$

The causation entropy $C_{\phi_n \to \dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]}$ is utilized to quantify the contribution from the candidate function ϕ_n to the dynamics $\dot{x}_i$ (i.e., the time derivative of the i-th state variable: x_i) conditioned on the remaining candidate functions $\boldsymbol{\Phi} \setminus \phi_n$, namely all the candidate functions except ϕ_n. This causation entropy reflects the causal influence of ϕ_n to the dynamics $\dot{x}_i$, and we enforce $\xi_{in} = 0$ if the causation entropy is small. Repeating this procedure over all $n = 1, \ldots, N$ and $i = 1, \ldots, p$ to form the matrix $\boldsymbol{\Xi}$. As only a few candidate functions will have the actual causal influence on the dynamics, the matrix $\boldsymbol{\Xi}$ is expected to have a sparse structure. The causation entropy $C_{\phi_n \to \dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]}$ is defined as follow:

$$C_{\phi_n \to \dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]} = H(\dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]) - H(\dot{x}_i | \boldsymbol{\Phi}), \tag{3.4}$$

where $H(\cdot|\cdot)$ is the conditional entropy, which is defined as:

$$H(U|V) = \int_u \int_v p(u, v) \log(p(u|v)) dv du, \tag{3.5}$$

where $p(u, v)$ is the corresponding probability density function (PDF) that can be determined by a histogram from the time series assuming ergodicity. On the right-hand side of (3.4), the difference between the two conditional entropies indicates the information in $\dot{x}_i$ contributed by the specific function ϕ_n given the contributions from all the other functions in the library $\boldsymbol{\Phi}$. Thus, it tells if ϕ_n provides additional information to $\dot{x}_i$. It is worth highlighting that the causation entropy in (3.4) is fundamentally different from directly computing the correlation between $\dot{x}_i$ and ϕ_n, as the causation entropy also considers the influence of the other library functions. If both $\dot{x}_i$ and ϕ_n are caused by a common factor ϕ_m, then $\dot{x}_i$ and ϕ_n can be highly correlated. Yet, in such a case, the causation entropy $C_{\phi_n \to \dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]}$ will be zero as ϕ_n is not the causation of $\dot{x}_i$.

In practice, the conditional entropy in (3.5) can involve expensive high-dimensional integrals, which is computationally challenging [238]. Nevertheless, a Gaussian approximation of the PDFs inside the integrand can be utilized to calculate the causation entropy [20]. By approximating all the joint and marginal distributions as Gaussians, the causation entropy is calculated as follows:

$$\begin{aligned} C_{W \to U|V} &= H(U|V) - H(U|V, W) \\ &= H(U, V) - H(V) - H(U, V, W) + H(V, W) \\ &\approx \frac{1}{2}\ln(\det(\mathbf{R}_{UV})) - \frac{1}{2}\ln(\det(\mathbf{R}_{V})) - \frac{1}{2}\ln(\det(\mathbf{R}_{UVW})) + \frac{1}{2}\ln(\det(\mathbf{R}_{VW})), \end{aligned} \tag{3.6}$$

where $\mathbf{R}$ denotes the covariance matrix of the corresponding vector, e.g., $\mathbf{R}_{UVW}$ corresponds to the covariance matrix of the vector $[U, V, W]^\top$. It should be noted that (3.6) employs general symbols of U, V, and W to indicate the calculation and the approximation of causation entropy. In terms of the connection to the system defined in (3.3), U corresponds to $\dot{x}_i$, W represents ϕ_n (i.e., the n-th basis function), and V is $[\boldsymbol{\Phi} \setminus \phi_n]$ (i.e., the basis functions excluding ϕ_n). The dynamics $\dot{x}_i$ can either be measured directly or obtained by numerical evaluating the temporal derivative of $x(t)$. In this paper, the latter method is adopted, and a fine temporal resolution data has been used to avoid the numerical discretization error caused by coarse temporal resolution. It should be noted that the numerical approximation of the temporal derivative is a non-trivial task. A common practice is to denoise the data before approximating the temporal derivative as suggested by [105]. If the actual frequency of the noises is much higher than the one of the signal, it is possible to have a robust numerical approximation with sub-sampled data. The calculation of causation

entropy is robust to the large but temporally uncorrelated noises which is demonstrated in Section 3.6.3. The explicit expression in (3.6) based on the Gaussian approximation can efficiently compute the causation entropy. It allows the computation of the causation entropy with a moderately large dimension, which is typically the case for many practical situations. For a p dimensional system and N basis functions, we need to calculate $p \times N$ causation entropies to formalize the final causation entropy matrix. The time complexity of causation entropy computing can be reduced by parallel computing and localized basis functions. Since the computation of the causation entropy for the dynamics of each state variable is independent with each other, we can calculate the causation entropy for different dynamics in parallel. The localized basis is a technique of dictionary design which constructs basis functions for each state using itself and its adjacent states. For example, each state variable x_i will only expanded with $\phi(x_i, x_{i\pm1}, ...x_{i\pm s})$ with s be a small number. For dynamical systems whose state variables only interact with the nearby ones, the technique of localized basis functions could be utilized to decrease N and reduce the time complexity of the causation entropy computation. It is worth noting that the Gaussian approximation may lead to certain errors in computing the causation entropy if the actual distribution is highly non-Gaussian. Nevertheless, the primary goal is not to obtain the exact value of the causation entropy. Instead, it suffices to detect if the causation entropy $C_{\phi_n \to \dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]}$ is nonzero (or practically above a small threshold value). In most applications, if a significant causal relationship is detected in the higher-order moments, it is very likely in the Gaussian approximation. This allows us to efficiently determine the sparse model structure. The exact values of the nonzero coefficients on the right-hand side of the identified model will be calculated via a simple least square estimation to be discussed in the following. Note that the Gaussian approximation is taken directly from the statistics associated with the time series from the underlying nonlinear model. Therefore, the Gaussian approximation still includes the nonlinear dynamical information. It is very different from linearizing a nonlinear complex system and computing the resulting Gaussian distribution. Such a Gaussian approximation of the time series has been widely applied to compute various information measurements and lead to reasonably accurate results [11, 239–241]. Note that, with linear and Gaussian assumptions, causation entropy has been demonstrated as equivalent to Granger causality [242], which has been a popular tool for analyzing time series data for decades [243]. But the focus here is more toward identifying the nonlinear models.

Below, for notation conciseness, $\mathbf{C}_{in}$ is utilized as a short-hand notation of $C_{\phi_n \to \dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]}$. To impose sparsity into $\boldsymbol{\Phi}$ in the practical computational scenarios, a threshold $\overline{C}$ is prescribed and $\mathbf{C}_{in} = 0$ is enforced when $C_{\phi_n \to \dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]} \leqslant \overline{C}$. Such a threshold value is adopted mainly to exclude the small causation entropy values due to the sampling error from using a finite time series. After applying this threshold value, a causation entropy matrix (CEM) $\mathbf{C}$ is obtained, where its (i, n)-th entry is given by:

$$\mathbf{C}_{in} = \begin{cases} 0 & \text{if } C_{\phi_n \to \dot{x}_i | [\boldsymbol{\Phi} \setminus \phi_n]} \leqslant \overline{C}, \\ 1 & \text{otherwise.} \end{cases} \tag{3.7}$$

Based on the matrix $\mathbf{C}$, the sparsity can be further enforced into $\boldsymbol{\Xi}$ by setting $\xi_{in} = 0$ when $\mathbf{C}_{in} = 0$. It has been demonstrated that a correct sparse model $\boldsymbol{\Xi\Phi}$ can be obtained in the offline-learning setting where the time series is long enough [20]. However, in the online learning setting, all the covariance matrices in (3.6) are only estimated from a limited amount of batch data and may not provide correct information for imposing sparsity. To address this challenge in the online learning setting, the following CEBoosting algorithm is introduced.

3.4 Causation Entropy Boosting for Online System Identification

Considering the data are sequential batches with the k-th batch data corresponds to a time series $\mathbf{x}(t)$ for $t \in [t_{B_k}, t_{B_{k+1}})$. Assuming that by exploiting all the past batch data $\mathbf{x}(t)$ for $t \in [0, t_{B_1})$, the causation entropy of the system has been calculated in an off-line mode, and then the model structure and unknown parameters in the current regime can be estimated based on the calculated causation entropy matrix. More specifically, the model in the current regime is estimated as:

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \end{bmatrix} = \begin{bmatrix} \xi^{(0)}_{1,1} & \cdots & \xi^{(0)}_{1,N} \\ \xi^{(0)}_{2,1} & \cdots & \xi^{(0)}_{2,N} \\ \vdots & \ddots & \vdots \\ \xi^{(0)}_{p,1} & \cdots & \xi^{(0)}_{p,N} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix} = \boldsymbol{\Xi}^{(0)} \boldsymbol{\Phi}. \tag{3.8}$$

With the 1-st new incoming batch data $\mathbf{x}(t)$ for $t \in [t_{B_1}, t_{B_2})$, the true dynamics $\dot{\mathbf{x}}$ within this period can be obtained by numerical differentiation of $\mathbf{x}(t)$. On the other hand, the predicted dynamics using the current model (3.8) is given by $\tilde{\dot{\mathbf{x}}} = \boldsymbol{\Xi}^{(0)}\boldsymbol{\Phi}$. Then the observed residual for the first batch is given by $\mathbf{r} = \dot{\mathbf{x}} - \tilde{\dot{\mathbf{x}}}$, which can be interpreted as a noisy observation of the residual dynamics $\delta\mathbf{f}$, i.e., $\mathbf{r} = \delta\mathbf{f} + \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ represents stochastic dynamical noise and errors arising from numerical differentiation. Therefore, the true dynamics is the temporal derivative from the new batch of time series, the predicted dynamics corresponds to the current model, and the residual dynamics is the difference between the true and predicted dynamics. If there is no regime switching for this batch, then it is expected that $\delta\mathbf{f} = 0$ and the estimated $\mathbf{r}$ is approximately Gaussian white noise. As an analog to (3.2), the matrix form of the noisy residual dynamics $\mathbf{r}$ can be written as:

$$\mathbf{r} = \begin{bmatrix} r_1(t) \\ r_2(t) \\ \vdots \\ r_p(t) \end{bmatrix} = \begin{bmatrix} \delta\xi_{1,1} & \cdots & \delta\xi_{1,N} \\ \delta\xi_{2,1} & \cdots & \delta\xi_{2,N} \\ \vdots & \ddots & \vdots \\ \delta\xi_{p,1} & \cdots & \delta\xi_{p,N} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix} + \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_N \end{bmatrix} = \delta\boldsymbol{\Xi}\boldsymbol{\Phi} + \boldsymbol{\eta}. \tag{3.9}$$

To detect regime switching, identify sparse pattern of $\delta\boldsymbol{\Xi}$ and estimate the rest of parameters of $\delta\boldsymbol{\Xi}$ in (3.9), one can estimate the causation entropy $C_{\phi_n \to r_i | [\boldsymbol{\Phi} \setminus \phi_n]}$ between basis of states $\boldsymbol{\Phi}$ and observations of residual dynamics $\mathbf{r}$ according to (3.4) and (3.6) and then obtain the corresponding causation entropy matrix $\mathbf{C}$ by (3.7).

Mathematically, if $\mathbf{C} = \mathbf{0}$, it implies that no variable in the basis functions $\boldsymbol{\Phi}$ is significant to the residual dynamics $\mathbf{r}$, which leads to $\delta\boldsymbol{\Xi} = \mathbf{0}$ in (3.9). In other words, the current model $\boldsymbol{\Xi}^{(0)}\boldsymbol{\Phi}$ fits the dynamics of 1-st batch and make residual dynamics $\mathbf{r}$ behave like white noise. This means there is no regime switching ($\delta\mathbf{f} = \mathbf{0}$) if $\mathbf{C} = \mathbf{0}$ and the model is unchanged. Otherwise, if $\mathbf{C}$ contains some non-zero terms, then a regime switching occurs ($\delta\mathbf{f} \neq \mathbf{0}$), and the goal is to identify the sparse structure of the $\delta\boldsymbol{\Xi}$ in (3.9). However, despite the mathematical justification, with a limited amount of data in each batch, the true sparse structure of $\delta\mathbf{f}$ cannot be accurately identified. It is often the case that $\mathbf{C}$ contains multiple entries that are nonzero due to the sampling error from the short time series. The sampling error can originate from insufficiently sampled data within a single data batch or measurement noises during the data collection process such as sensor-based data. Therefore, when a batch of time series is short, the sampling error will cause a biased

statistical estimation of CEM. This means if directly imposing the sparsity into $\delta\boldsymbol{\Xi}$ based on the CEM $\mathbf{C}$ computed from such a short single-batch data may lead to an incorrect residual system $\delta\boldsymbol{\Xi}\boldsymbol{\Phi}$. To resolve such a sampling problem, we introduce aggregated CEM $\mathbf{C}^+(K)$ based on the average of the causation entropy from a series of data batches. The (i,n)-th entry of $\mathbf{C}^+(K)$ is given by:

$$\mathbf{C}^+(K)_{in} = \begin{cases} 0 & \text{if } \frac{1}{K}\sum_{k=1}^{K} C^{(k)}_{\phi_n \to r_i | [\boldsymbol{\Phi}\setminus\phi_n]} \leqslant \overline{C}, \\ 1 & \text{otherwise.} \end{cases} \tag{3.10}$$

Denote by D the number of batches with which the aggregated CEM has not changed, namely

$$\mathbf{C}^+(K) = \mathbf{C}^+(K-d), \quad \text{for all } d = 1, 2, \ldots, D-1. \tag{3.11}$$

In the CEBoosting algorithm, D is a hyper-parameter and needs to be pre-determined. The algorithm is robust with the choice of D as long as D is not too small. If the averaged causation entropy matrix with a chosen value of D can successfully identify all the important basis functions, then any larger value of D should also complete the identification task with the consistent pattern of important basis functions. Based on our tests of all the examples, $D = 4$ is a good empirical choice to achieve the consistent and correct pattern of identified basis functions. We further define a stable aggregated causation entropy matrix $\overline{\mathbf{C}^+} = \mathbf{C}^+(K^*)$ with the smallest K^* that satisfies the criterion in (3.11).

With this stable aggregated causation entropy matrix $\overline{\mathbf{C}^+}$, we then impose sparsity into $\delta\boldsymbol{\Xi}$ by setting $\delta\xi_{in} = 0$ when $\overline{\mathbf{C}^+}_{in} = 0$ and estimating a set of remaining coefficients $\delta\boldsymbol{\Xi} = \{\delta\xi_{in} | \overline{\mathbf{C}^+}_{in} = 1\}$.

Once the sparsity is imposed into $\delta\boldsymbol{\Xi}$, the model $\delta\boldsymbol{\Xi}\boldsymbol{\Phi}$ in (3.9) can be calibrated based on the accumulated batches of data $\mathbf{x}(t)$ for $t \in [t_{B_1}, t_{B_{K^*+1}})$. Assume a discrete approximation of the continuous data with a fixed time step Δt such that $t_{B_{k+1}} - t_{B_k} = M\Delta t$ for any k. The model calibration is performed by solving the following least squares problem:

$$\arg\min_{\delta\boldsymbol{\Xi}} \sum_{m=0}^{MK^*-2} \|\mathbf{r}(t_{B_1} + m\Delta t) - \delta\boldsymbol{\Xi}\boldsymbol{\Phi}(t_{B_1} + m\Delta t)\|^2, \tag{3.12}$$

where $\|\cdot\|$ denotes the vector norm in $\mathbb{R}^p$. Note that the method also works with adaptive time steps, and the assumption of a fixed time step in (3.12) is for the simplicity of the

illustration.

After the sparse parameter matrix $\delta\boldsymbol{\Xi}$ of residual model (3.9) is obtained, the current parameter matrix $\boldsymbol{\Xi}^{(0)}$ is updated by adding $\delta\boldsymbol{\Xi}$, which is the parameter matrix of residual model. The CEBoosting algorithm repeats the above procedure when a new batch of data arrives. A schematic illustration of Lorenz 63 system with regime switching is displayed in Figure 3.1, and more details of the CEBoosting algorithm can be found in Algorithm 4 presented in Appendix B. It should be noted that the notations adopted in this section assume the regime switching time $t_s \in [t_{B_1}, t_{B_2})$ for the simplicity of the illustration. In practice, regime switching can happen at $t_s \in [t_{B_k}, t_{B_{k+1}})$ with $k > 1$, for which Algorithm 4 presents the detailed procedures of detecting regime switching, aggregating causation entropy matrix, identifying a sparse model structure and then fitting the model parameters.

In recent years, SINDy has been a popular framework for sparsely identifying nonlinear dynamics from data [105]. SINDy exploits an iterative thresholding regularization to determine the model structure and estimate the model parameters simultaneously. The iterative thresholding regularization guarantees the parsimonious model structure. Using abundant training data (e.g., long time-series), the original SINDy method was designed for offline model identification. To better work with a limited amount of data, a recent extension of SINDy [196] leveraged the statistical approach of bagging and achieved a more robust learning performance. Compared to the SINDy methods, the CEBoosting algorithm is mainly designed with a low computational cost for robust online learning with limited sequential data. The key difference from the SINDy methods is that imposing sparsity is decoupled from the parameter estimation in the CEBoosting algorithm. Specifically, causation entropy is utilized only to determine a parsimonious model structure without dealing with the model calibration. The concept of bagging is further introduced with sequential batches of data to ensure robust estimation of causation entropy before imposing a parsimonious model structure. It results in utilizing the minimum data to determine the model structure. With a robust estimation of the parsimonious model structure, the CEBoosting algorithm exploits a simple least square estimate to solve the parameter values, a quadratic optimization problem with a closed analytic solution. This avoids repeatedly estimating the parameters in the LASSO-type or thresholding-based regression approaches when determining the model structure by examining each batch data. In online learning, the CEBoosting algorithm, by design, can also avoid accidentally removing essential basis functions due to the incorrectly calibrated model based on a limited amount of data.

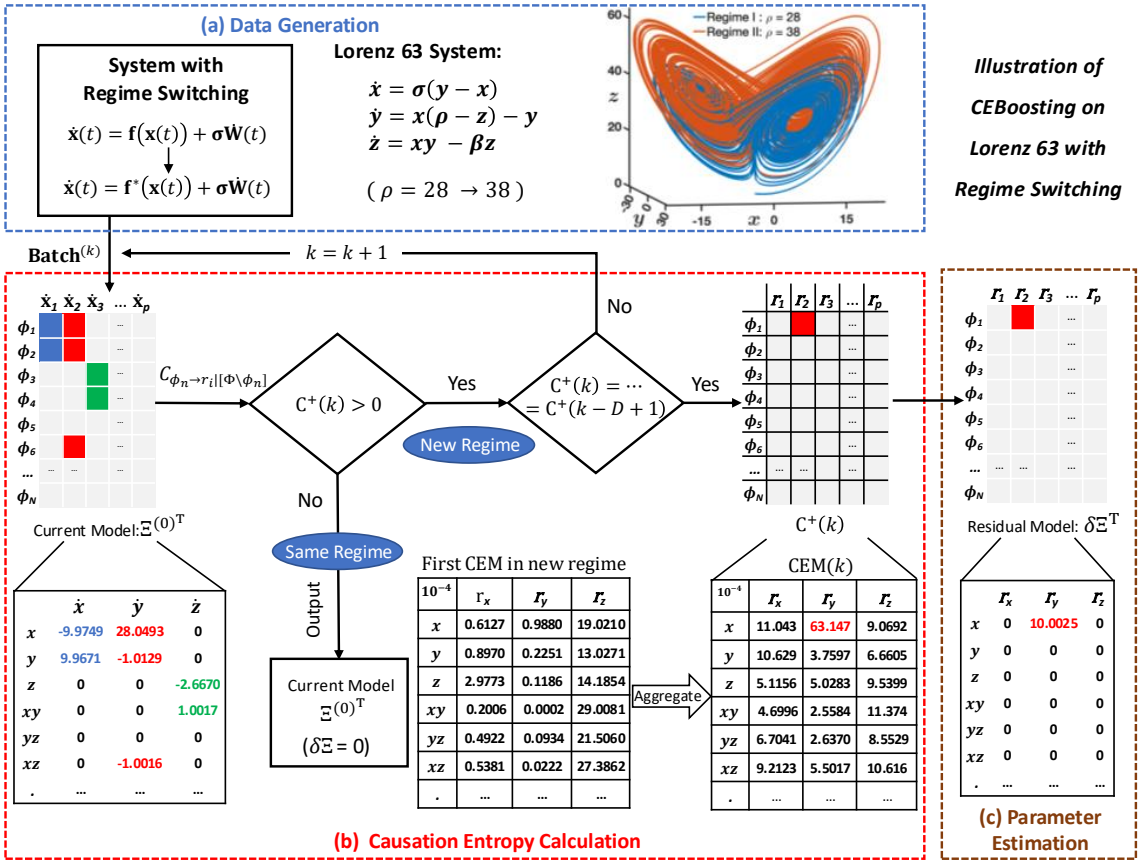

Figure 3.1: Schematic of CEBoosting algorithm (based on the Lorenz 63 model). Panel (a): data is generated from the Lorenz 63 system with regime switching. The parameter in Regime 1 is $\sigma = 10, \beta = 8/3, \rho = 28$, while ρ is changed to 38 in Regime 2. The index of incoming batch data k starts with 1. Panel (b): $\boldsymbol{\Xi}^{(0)}$ is the current model parameter matrix defined in (3.3) with $\boldsymbol{\Phi}$ is the basis functions and $\dot{x}_i$ is the dynamic of state x_i. With $\boldsymbol{\Xi}^{(0)}$, the residual dynamics r_i and the causation entropy between r_i and $\boldsymbol{\Phi}$ can be calculated for each batch. CEM(k) is the aggregated causation entropy from batch 1 to k. $\mathbf{C}^+(k)$ is the binary matrix of CEM(k) defined in (3.10) indicating the sparse structure of the residual model. D is defined in (3.11) indicating number that $\mathbf{C}^+(k)$ becomes stable, i.e., the pattern is consistent with the previous D aggregated causation entropy matrix. Panel (c): with $\mathbf{C}^+(k)$ and data batches from the batch with new regime detected to the one with a stable $\mathbf{C}^+$, the residual model is calibrated by least square estimation.

3.5 Application of CEBoosting with Data Assimilation

The CEBoosting algorithm can be extended to scenarios with partial observations. In many practical applications, observations are not available for hidden state variables or parameters. In this section, with appropriate incorporation of data assimilation, CEBoosting can be directly applied under such sequential partial observation settings for identifying underlying changing of hidden states in dynamcial systems.

Considering the dynamical system in (3.13) which is parameterized by hidden states $\boldsymbol{\theta}$:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}) + \boldsymbol{\sigma}\dot{\mathbf{W}}(t), \\ \dot{\boldsymbol{\theta}}(t) &= \mathbf{g}(\boldsymbol{\theta}(t)) + \boldsymbol{\sigma}_{\boldsymbol{\theta}}\dot{\mathbf{W}}_{\boldsymbol{\theta}}(t), \end{aligned} \tag{3.13}$$

where $\boldsymbol{\theta} = [\theta_1(t), \theta_2(t), \ldots, \theta_J(t)]^\top \in \mathbb{R}^J$ represents the hidden states that drive the dynamics of the system state $\mathbf{x}$. Here, $\mathbf{g} : \mathbb{R}^J \mapsto \mathbb{R}^J$ is a vector-valued nonlinear function, $\dot{\mathbf{W}}_{\boldsymbol{\theta}}$ denotes white noise, and $\boldsymbol{\sigma}_{\boldsymbol{\theta}}$ represents the corresponding noise coefficients.

A regime switching in the evolution of $\mathbf{x}$ occurs when specific variables within the hidden states $\boldsymbol{\theta}$ are either removed from or augmented within the dynamical function $\mathbf{f}$, causing a transition to a new regime $\mathbf{f}^*$. Assuming the functional form of (3.13) is known, CEBoosting can be employed to detect shifting patterns in the parameters governing the dynamics of $\mathbf{x}$. This is achieved using estimates of $\boldsymbol{\theta}$ derived from observations of $\mathbf{x}$ via data assimilation. Detailed introduction for data assimilation—including filters, smoothers, and samplers—are provided in Appendix D.

Given sequential data for the state $\mathbf{x}$, data assimilation is first applied to estimate the trajectory of the hidden states $\boldsymbol{\theta}$. Subsequently, with the DA-estimated hidden states $\boldsymbol{\theta}$, a dictionary $\boldsymbol{\Phi} = [\phi_1, \phi_2, \ldots, \phi_N]^\top$ is constructed using the joint input vector $(\mathbf{x}, \boldsymbol{\theta})$, where each basis function is defined as $\phi_n := \phi_n(\mathbf{x}, \boldsymbol{\theta})$. CEBoosting is then utilized to calculate the aggregated causation entropy $\mathbf{C}^+(K)$ to identify significant basis functions. Based on the stability conditions introduced in previous section, the significant basis functions are selected. Any components of $\boldsymbol{\theta}$ that do not appear in any of the identified significant basis functions are determined to have no contribution and are removed from the dynamics of $\mathbf{x}$; otherwise, they are retained. By combining data assimilation and CEBoosting, we can effectively monitor the evolving behavior of hidden states $\boldsymbol{\theta}$ within the dynamical systems using only sequential observations of the system state $\mathbf{x}$.

In Section 3.6.4, a test case of stochastic parameterized model is shown to demonstarte the application of the CEBoosing with the data assimilation, which identify regime switching triggered by the unobserved hidden processes.

3.6 Numerical Experiments

In this section, the performance of the CEBoosting method is demonstrated utilizing four different chaotic or turbulent systems, which are models that mimic crucial features in many science or engineering disciplines. The presentation starts with the three-dimensional Lorenz 63 system, which is a classical chaotic system. This test is utilized as a proof of concept to demonstrate the detailed steps of the method in the online identification of the non-linear dynamics with regime switching. To illustrate the efficiency of the algorithm in capturing the regime switching behavior in a relatively high-dimensional case, the forty-dimensional Lorenz 96 system is utilized as a second test, where the localization technique is incorporated to mitigate the curse of dimensionality. As strongly non-Gaussian statistics, intermittency and extreme events appear in many climate, atmosphere and ocean science problems, a multi-mode layered topographic model that captures these crucial turbulent features is adopted as the next test model. The last test case aims to deal with a more realistic scenario where only a subset of the state variables is observed. Data assimilation is, therefore, essential in such a partial observational case. A stochastic parameterized extended Kalman filter (SPEKF) model is used to estimate and simulate the hidden system state variables. Below, assuming the starting model and its parameters are available for all the numerical examples. The goal is to learn the regime switching and the corresponding residual model that adjusts the original model to a new regime based on online sequential data.

For all four systems, the numerical simulation time step size is chosen as $\mathrm{dt} = 0.001$. The hyper-parameter D in (3.11) is chosen as 4 for the numerical examples of this work.

3.6.1 The Lorenz 63 System: A Classical Chaotic System

The Lorenz 63 (L63) system is proposed by Lorenz in 1963 [244]. It is a simplified mathematical model for atmospheric convection. The equations relate the properties of a two-dimensional fluid layer uniformly warmed from below and cooled from above. In particular,

the equations describe the rate of change of three quantities concerning time: x is proportional to the rate of convection, y to the horizontal temperature variation, and z to the vertical temperature variation. The constants σ, ρ, and β are system parameters proportional to the Prandtl number, Rayleigh number, and certain physical dimensions of the layer itself [245]. The L63 model is also widely used as a simplified model for lasers, dynamos, thermosyphons, brushless DC motors, electric circuits, chemical reactions, and forward osmosis [246–252]. The governing equation of the Lorenz 63 system is as follows,

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= x(\rho - z) - y, \\ \frac{dz}{dt} &= xy - \beta z. \end{aligned} \tag{3.14}$$

The standard parameters $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ that create the butterfly profile are utilized as the starting regime. The new regime takes a different value of the parameter $\rho = 38$. The sudden change of the parameter from $\rho = 28$ to $\rho = 38$ occurs at $t = 100$. The goal is to (i) detect the regime switching and (ii) learn the residual model that adjusts the original system to the new one.

Figure 3.2(a) compares the trajectories in the phase space between the original (Regime 1) and new (Regime 2) regimes. It can be seen that the two systems are on different manifolds in the phase space. Figure 3.2(c) shows the time series of the three state variables. With the regime switching at $t = 100$, it can be seen in Figure 3.2(c) that the patterns of time series change accordingly, especially for the variable z that demonstrates a shift of its mean value. The autocorrelation function (ACF) of each state variable for the original system is presented in Figure 3.2(b), which shows a rapid decay of correlation within one time unit, except for the variable z that has some oscillations in its ACF. Similar behavior of the ACF is observed in Figure 3.2(d) after the regime switching. Finally, Figure 3.2(e) shows the ensemble mean of the state variable z as a function of time from an independent simulation with 5000 ensemble members, which reveals that the transition time of the regime switching is about 20 time units. This also indicates that the transition time depends on the property of the transient feature and is very different from the decorrelation time. Nevertheless, the decorrelation time of the original system provides a natural way to determine the batch size. Thus, the batch size is chosen as one time unit here.

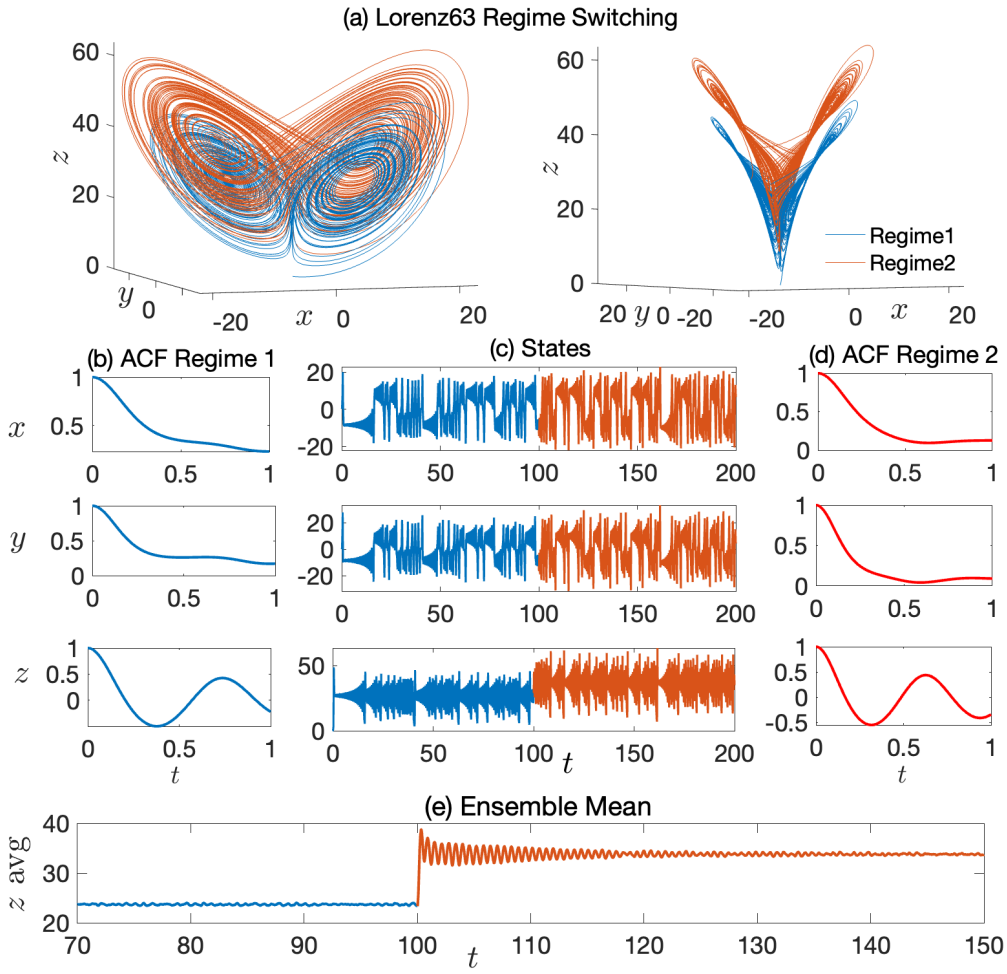

Figure 3.2: Lorenz 63 system with regime switching. (a): trajectories of the original system and the new one in phase space. (c): time series of system state variables. (b) and (d): autocorrelation function (ACF) of the original system and the new regime. (e): ensemble mean of variable z before and after regime switching at $\mathrm{t} = 100$.

The CEBoosting algorithm is employed to detect the regime switching and identify the sparse structure of the residual model. The candidate basis functions include all the linear and quadratic nonlinear functions: $\{x, y, z, xy, xz, yz, x^2, y^2, z^2\}$. With the identified sparse structure, the coefficients of the residual model are then determined via the least square estimation. To detect the regime switching and identify the sparse structure of the residual model, the aggregated causation entropy matrix is gradually updated until its structure gets stable, i.e., the causation entropy values of some candidate basis functions keep being significantly greater than others. The stable causation entropy matrix structure is obtained after 6 time units and is presented in Table 3.1, where only one entry has a significant value. It is worthwhile to highlight that the total time units (6 units) to discover the regime switching and determine the model parameters in the new regime is shorter than the transition time (20 units). In other words, utilizing the information from the transient period is sufficient for the CEBoosting algorithm to determine the model response to the regime switching in this test case.

Table 3.1: The L63 model - The causation entropy matrix (CEM) after 6 time units. The pattern of CEM becomes stable and does not change with incorporating more batch data. The entry with a significant value of the causation entropy is highlighted using the bold font. According to the pattern of this CEM, a residual model will be built.

10^{-4}	x	y	z	xy	xz	yz	x^2	y^2	z^2
$\dot{x}$	11.0437	10.6292	5.1156	4.6996	6.7041	9.2123	8.9308	3.0866	4.0888
$\dot{y}$	**63.1474**	3.7597	5.0283	2.5584	2.6370	5.5017	4.3689	2.4472	4.4099
$\dot{z}$	9.0692	6.6605	9.5399	11.3740	8.5529	10.6165	10.8299	10.8725	10.5916

Finally, based on the sparse structure identified in Table 3.1, a residual model can be calibrated via the least square estimation based on the residuals of the original model. The residual model is then added to the original model as a correction term. The coefficients of the corrected model for the new regime are listed in Table 3.2. It can be seen that the corrected model successfully captures the crucial parameter value $\rho = 38$ that induces the new regime.

Table 3.2: The L63 model - Updated model parameters for the new regime.

	x	y	z	xy	xz	yz	x^2	y^2	z^2
$\dot{x}$	-9.9749	9.9671	0	0	0	0	0	0	0
$\dot{y}$	**38.0357**	-1.0129	0	0	0	-1.0016	0	0	0
$\dot{z}$	0	0	-2.6670	1.0017	0	0	0	0	0

3.6.2 The Lorenz 96 System: Strategy for Applying CEBoosting to Relatively High-Dimensional Systems

The Lorenz 96 (L96) system is a classical chaotic to turbulent model [253]. It can be regarded as a coarse discretization of atmospheric flow on a latitude circle with complicated wave-like and chaotic behavior. It is also widely used as a testbed for data assimilation, prediction, and uncertainty quantification [254–257]. The L96 model reads:

$$\frac{dx}{dt} = (x_{j+1} - x_{j-2})x_{j-1} - x_j + F, \qquad j = 1, 2, ..., J, \tag{3.15}$$

with periodic boundary conditions $x_{-1} = x_{J-1}, x_0 = x_J, x_{J+1} = x_1$ and $J = 40$ is adopted. The L96 system is utilized here to demonstrate that the CEBoosting algorithm, combined with localization techniques, can detect regime switching for relatively high-dimensional systems. The motivation here is that the number of candidate functions quickly increases with the dimension of the system if all possible combinations of state variables up to a particular order are included. As a result, the computational cost becomes unaffordable without suitable treatment for such a curse of dimensionality. To this end, the idea of localization is exploited here. Localization means the dynamics of each state variable only depend on the nearby ones. In fact, the advection, diffusion, and dispersion are all local operators [10, 102]. Similarly, the localization applies to many parameterization problems for the subgrid scales, which also depend only on the nearby corresponding large-scale state variables [258–260]. The idea of localization is also widely utilized in data assimilation and prediction [261–263]. Localization reduces the number of candidate functions for the dynamics of each x_j by including only the terms that represent local interactions with dx_j/dt. Note that the total number of candidate functions in the entire library can remain large, but only a relatively small number of functions will be examined for the causal relationship to the dynamics of each x_j.

The original system (Regime 1) has a forcing coefficient $F = 8$, which makes the system have a strongly chaotic behavior. After the regime switching at $t = 100$, the new system (Regime 2) takes the forcing value $F = 16$. Meanwhile, the coefficient of x_j, representing the damping effect, changes from -1 to -1.5. This leads to a fully turbulent regime [64]. Some weak coherent structures can still be observed in the strongly chaotic regime, but they disappear in the fully turbulent one. See Figure 3.3 for the two regimes. Figure 3.4 shows the detailed statistical properties of the state variable x_{10}. Note that the model has homogeneous dynamics, so the statistical properties for different state variables are the same. Figure 3.4(a) illustrates how the time series of x_{10} change with the regime switching. From the strongly chaotic to the fully turbulent regime, the variance of all the state variables becomes more extensive, and the decorrelation time becomes shorter. The batch size of 1 time unit is chosen here, which is again of the same order as the decorrelation time. The total transition time is about 1 to 2 units, according to Figure 3.4(b).

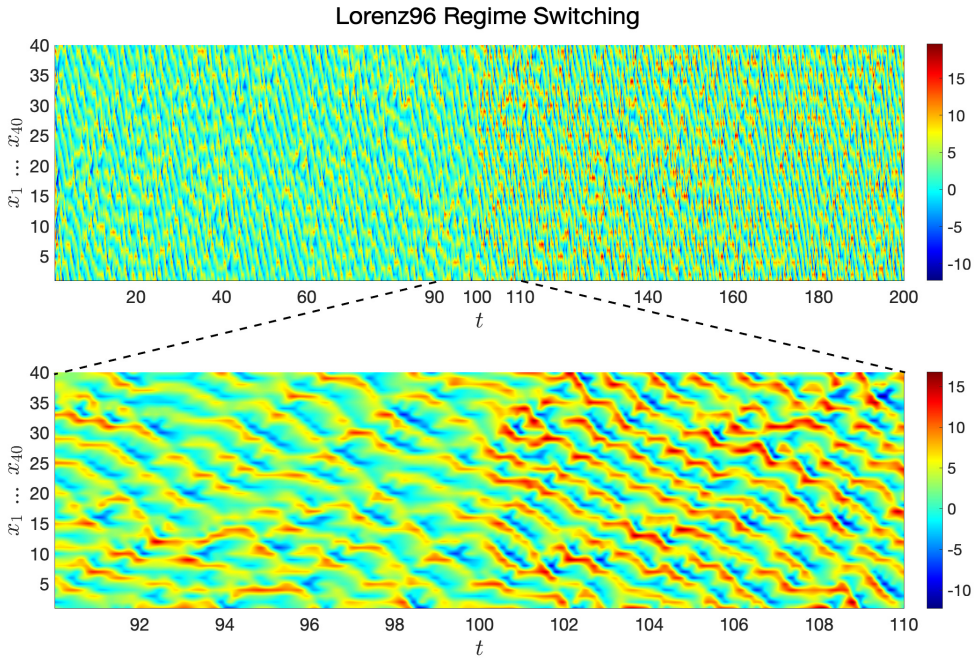

Figure 3.3: The Lorenz 96 system with regime switching. Top: forty-dimensional system state evolving at the time interval $[0, 200]$ with a regime switching at $t = 100$. Regime 1 is chaotic with $F = 8$, and regime 2 is turbulent with $F = 16$ and the linear terms $-1.5x_j$. Bottom: zoom-in view of the regime switching at the time interval $[90, 110]$.

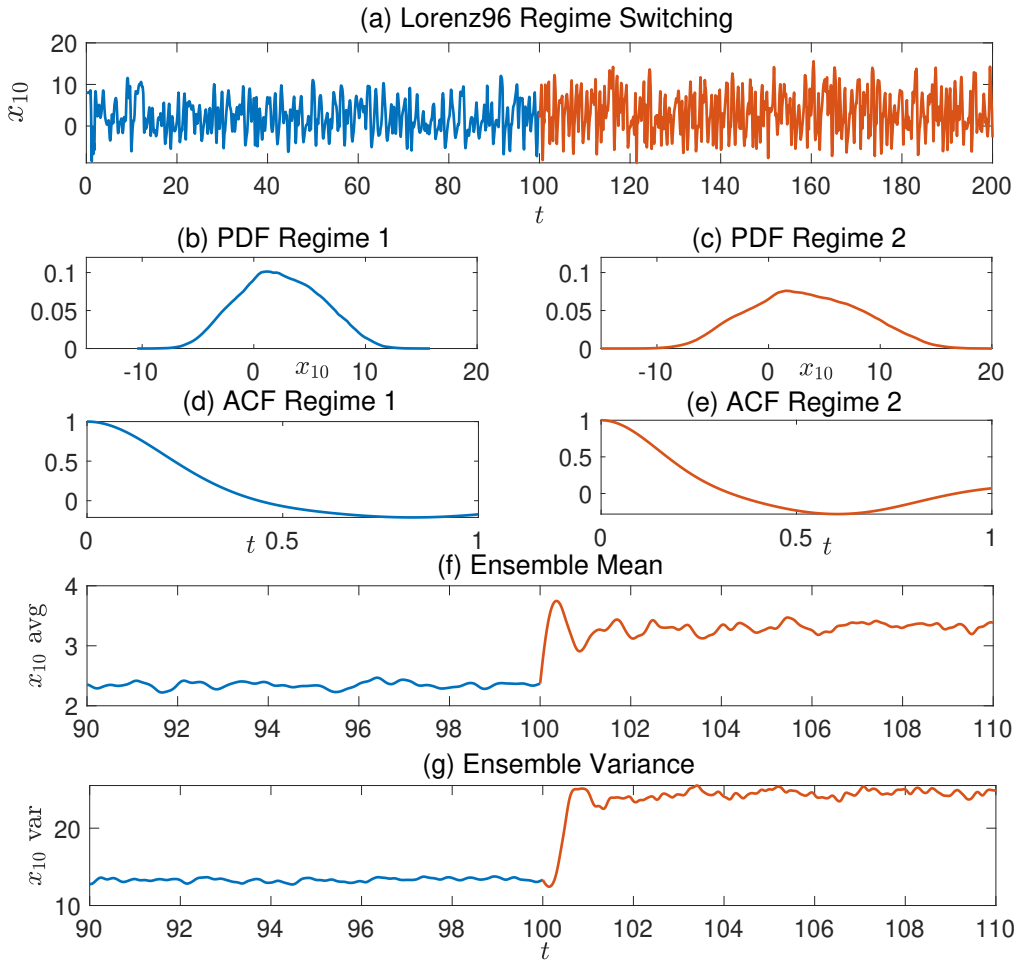

Figure 3.4: Statistical properties of the state variable x_{10} of the Lorenz 96 system with regime switching. Panel (a): time series of x_{10} in regime 1 and regime 2. Panels (b) and (c): the PDFs of x_{10} in both regimes. Panels (d) and (e): the ACFs of x_{10} in both regimes. Panels (f) and (g): the ensemble mean and the ensemble variance of x_{10} before and after the regime switching at $t = 100$.

A set of candidate basis functions $\{x_j, x_j^2, x_j x_{j-1}, x_j x_{j+1}, x_j x_{j-2}, x_j x_{j+2}\}$ is built by exploiting polynomials up to the second order for each state x_j, $j = 1, 2, ...J$. Note that the residual dynamics of each state x_j is assumed to have contributions from the basis functions that consist of only its adjacent states. The CEBoosting algorithm identifies the sparse structure of the residual model using about two batches of data (i.e., the causation entropy matrix pattern does not change after two time units). Table 3.3 shows the CEM based on utilizing two batches of data, which is about the same length as the estimated transition time of Lorenz 96 system (see Figure 3.4(f-g)). It is seen that the causation entropy entries associated with the actual residual terms are at least one order more significant than others. Therefore, the CEBoosting algorithm successfully identifies the correct sparse structure of the residual model based on data within the time interval of the transient period.

Table 3.3: The L96 model - The CEM after 2 time units. The entries with significant values of the causation entropy are highlighted using the bold font. The pattern of CEM becomes stable and does not change with incorporating more data batches. According to the pattern of this CEM, a residual model will be built.

10^{-4}	x_1	x_1^2	x_2x_{40}	$x_{39}x_{40}$	x_2	x_2^2	x_3x_1	$x_{40}x_1$	$\cdots$	x_{40}	x_{40}^2	x_1x_{39}	$x_{38}x_{39}$
$\dot{x}_1$	**436.0597**	4.2015	2.2964	1.4823	0.0976	3.4716	1.3792	0.9251	$\cdots$	0.1869	3.0008	0.7236	8.0802
$\dot{x}_2$	8.4920	9.8888	6.0428	8.0793	**19.8015**	8.3095	10.1492	9.6844	$\cdots$	0	0	0	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\dot{x}_{40}$	2.7772	3.1363	4.5714	8.4405	9.6327	5.3498	4.8701	7.6239	$\cdots$	**200.3710**	2.5200	7.6225	2.7748

After obtaining the sparse structure according to the CEM in Table 3.3, the residual dynamics are calibrated via a linear combination of candidate basis functions, and the coefficients for the linear combination is obtained by the least square estimation. The L96 model in the new regime is then updated by adding this residual model to the original model. The coefficients of the updated model are summarized in Table 3.4. The identified model shows a good agreement with the true system of the new regime with $F = 16$ and the linear term $-1.5x_j$.

3.6.3 The Topographic Model: Dynamical System with Intermittency and Extreme Events

The topographic model is an ideal model to study the complex nonlinear interaction of the large-scale and the small-scale flow and the role of the topography [3, 264]. The topographic

Table 3.4: The L96 model - Coefficients of the updated model for the new regime. The entries with bold font indicate the parameters that contribute to the regime switching.

	1	x_1	x_1^2	x_2x_{40}	$x_{39}x_{40}$	x_2	x_2^2	x_3x_1	$x_{40}x_1$	$\cdots$	x_{40}	x_{40}^2	x_1x_{39}	$x_{38}x_{39}$
$\dot{x}_1$	**16.0248**	**-1.4938**	0	1.0000	-1.0016	0	0	0	0	$\cdots$	0	0	0	0
$\dot{x}_2$	**16.0490**	0	0	0	0	**-1.5077**	0	1.0000	-1.0000	$\cdots$	0	0	0	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\dot{x}_{40}$	**16.0027**	0	0	0	0	0	0	0	0	$\cdots$	**-1.5017**	0	1.0000	-1.0000

model can generate intermittency and extreme events. The corresponding PDF is often highly non-Gaussian with heavy tails. Therefore, it provides a challenging test case for detecting the regime switching behavior. Here the small-scale flow is given in terms of the stream function ψ. The large-scale velocity field only has the zonal component $u(t)$, and the topography is given by the function h. The parameter $\beta > 0$ is the contribution from the beta-plane effect. A common simplified version of the topographic model assumes a layered topography along the y direction, which means ψ is only a function of y. In addition, the model contains only the leading two Fourier wavenumbers of the stream function (with $k = \pm 1$ and ± 2). With these simplifications, the resulting model is reduced to a 5 dimensional system containing u, $\psi_{\pm 1}$ and $\psi_{\pm 2}$. For the simplicity of notation, a change of variables defines the new state variables $v_1, \ldots, v_4$, which are linked with $\psi_{\pm 1}$ and $\psi_{\pm 2}$ via

$$\psi_1 = \frac{1}{2\sqrt{2}}\left((v_2 - v_1) - (v_2 + v_1)i\right) \qquad \text{and} \qquad \psi_2 = \frac{1}{2\sqrt{2}}\left((v_4 - v_3) - (v_4 + v_3)i\right)$$

where i is the imaginary unit. Similarly, ω_1 and ω_3 are the two new variables standing for the Fourier coefficients of the topographic effect from h. The model reads:

$$\begin{aligned}
\frac{dv_1}{dt} &= -d_{v_1}v_1 - \beta v_2 + v_2 u - 2\omega_1 u + \sigma_{v_1}\dot{W}_1 \\
\frac{dv_2}{dt} &= -d_{v_2}v_2 - \beta v_1 - v_1 u + \sigma_{v_2}\dot{W}_2 \\
\frac{dv_3}{dt} &= -d_{v_3}v_3 - \omega_3 u - \frac{\beta}{2}v_4 + 2v_4 u + \sigma_{v_3}\dot{W}_3 \\
\frac{dv_4}{dt} &= -d_{v_4}v_4 - \frac{\beta}{2}v_3 - 2v_3 u + \sigma_{v_4}\dot{W}_4 \\
\frac{du}{dt} &= -d_u u + \omega_1 v_1 + 2\omega_3 v_3 + \sigma_u \dot{W}_u
\end{aligned} \tag{3.16}$$

The original system (Regime 1) takes the parameters $d_{v_1} = d_{v_2} = d_{v_3} = d_{v_4} = d_{v_5} = 0.005$, $\beta = 1$, $\sigma_{v_1} = \sigma_{v_2} = \sigma_{v_3} = \sigma_{v_4} = 1/(20\sqrt{2})$, $\sigma_u = 1/\sqrt{2}$, $\omega_1 = \sqrt{2}/2$ and $\omega_3 = \sqrt{2}/4$. The new model (Regime 2) utilizes different parameters for the topographic effect with $\omega_1 = \omega_3 = 3\sqrt{2}/2$. It is worthwhile to note that the noise level in the dynamics of the zonal flow u is much larger than those in the dynamics of v_i in (3.16). This is a typical situation as the u is the only mode that explicitly describes the zonal feature of the flow. A large noise is taken to mimic the unresolved dynamical features.

Figure 3.5 displays the regime switching and statistical properties of all five variables in the topographic model. The pattern and amplitude of each time series demonstrate a noticeable change after the regime switching happens at $t = 2500$. Note that the long time series of both regimes are utilized here, ensuring that the associated statistics are computed with a sufficiently large number of sample points. The PDFs behave like Laplace distributions with heavy tails, indicating many extreme events in the model simulation. The ACF of u in the original model (regime 1) decays very slowly, taking about 30 time units until the ACF approaches zero. Because of this, a batch size of 30 time units is utilized in the CEBoosting algorithm. Due to the strengthening of the topographic effect after the regime switching, the time series of v_2 and v_4 in the new model (regime 2) display multiscale features, where the ACFs have a quick decay at the beginning but then relax to zero slowly after 50 time units.

Similar to the previous test models, the library contains the polynomials up to the second-order. In other words, the following twenty basis functions are employed to build the library: $\{v_1, v_2, v_3, v_4, u, v_1^2, v_2^2, v_3^2, v_4^2, u^2, v_1v_2, v_1v_3, v_1v_4, v_1u, v_2v_3, v_2v_4, v_2u, v_3v_4, v_3u, v_4u\}$.

As the noise levels of state variables are significantly different from each other, the magnitudes of causation entropy for different residual dynamics are not the same either. Therefore, the selection of the candidate functions is based on the causation entropy values for the dynamics of each state variable separately. To determine the entire residual model, 60 time units is used. See Table 3.5.

Based on the sparse model structure identified in Table 3.5, the residual model is calibrated via the least square estimation. The coefficients of the calibrated model for the new regime are summarized in Table 3.6. It can be seen that the corrected model successfully updates the changed parameters ω_1 and ω_3 in the new regime. The resulting model can reproduce the strong non-Gaussian features with intermittency and extreme events.

It is worthwhile to remark that, as the noise levels in the dynamics of v_i are lower than

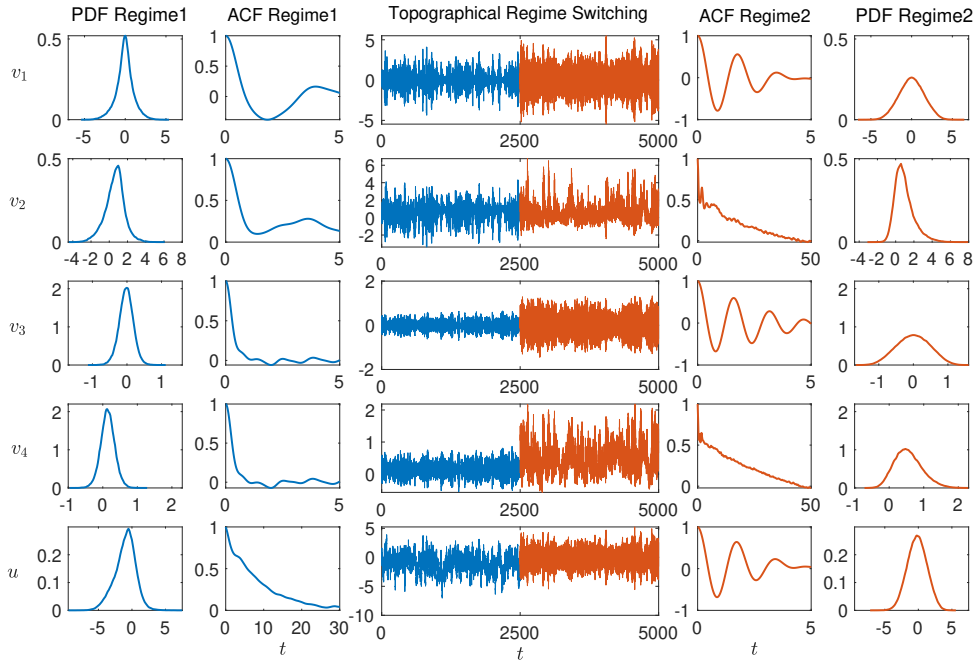

Figure 3.5: Topographic model with regime switching. Middle column: time series of each state in the time interval [0, 5000] with regime switching at $t = 2500$. 1st and 5th columns: probability density function (PDF) of each state in both regimes. 2nd and 4th columns: autocorrelation function (ACF) of each state in both regimes.

Table 3.5: The topographic model - The CEM after 60 time units. The pattern of CEM does not change with incorporating more batch data. The entries with significant values of the causation entropy are highlighted using the bold font.

10^{-3}	v_1	v_2	v_3	v_4	u	v_1u	v_2u	v_3u	v_4u	$\cdots$
$\dot{v}_1$	0.0115	0.0227	0.0165	0.0614	**120.9879**	0.0216	0.0365	0.0407	0.0249	$\cdots$
$\dot{v}_2$	0.0124	0.0327	0.0066	0.0150	0.0313	0.0002	0.0090	0.0039	0.0284	$\cdots$
$\dot{v}_3$	0.0023	0.0033	0.0142	0.0099	**51.3949**	0.0537	0.0291	0.0572	0.0082	$\cdots$
$\dot{v}_4$	0.0359	0.0619	0.0150	0.0134	0.0253	0.0180	0.0127	0.0099	0.0210	$\cdots$
$\dot{u}$	**0.1080**	0.0022	**0.0471**	0.0146	0.0065	0.0068	0.0097	0.0049	0.0063	$\cdots$

Table 3.6: The topographic model - Updated model for the new regime

	ν_1	ν_2	ν_3	ν_4	u	$\nu_1 u$	$\nu_2 u$	$\nu_3 u$	$\nu_4 u$	$\cdots$
$\dot{\nu}_1$	-0.0496	-1.0008	0	0	**-4.2463**	0	0.9998	0	0	$\cdots$
$\dot{\nu}_2$	0.9988	-0.0501	0	0	0	-1.0009	0	0	0	$\cdots$
$\dot{\nu}_3$	0	0	-0.0554	-0.4968	**-2.1200**	0	0	0	2.0015	$\cdots$
$\dot{\nu}_4$	0	0	0.4998	-0.0523	0	0	0	-1.9996	0	$\cdots$
$\dot{u}$	**2.1123**	0	**4.5713**	0	-0.0599	0	0	0	0	$\cdots$

that in u, a shorter time series (and shorter batch length) with in total of only 15 time units can be utilized to reach a stable CEM for the ν_i components. See Table 3.7. Figure 3.6 shows the time evolution of the ensemble mean and the ensemble variance of the topographic model, including the time instant of regime switching. The results here can be utilized to infer the transition time, which is about 20 time units. Therefore, in this topographic model with a large noise in the u dynamics, the stable pattern of all ν_i variables can be identified by the CEBoosting algorithm within the transition period. Yet, 60 time units of data are needed for identifying the model structure associated with the variable u, mainly due to the larger noises of u.

Table 3.7: The topographic model - The CEM after 15 time units. The pattern of all ν variables is stable. The entries with significant values of the causation entropy are highlighted using the bold font.

10^{-3}	ν_1	ν_2	ν_3	ν_4	u	$\nu_1 u$	$\nu_2 u$	$\nu_3 u$	$\nu_4 u$	...
$\dot{\nu}_1$	0.0182	0.0233	0.0322	0.0679	**22.665**	0.0244	0022	0.1726	0.0285	...
$\dot{\nu}_2$	0.0768	0.0071	0.0002	0.0004	0.0879	0.0582	0.0007	0.0171	0.1059	...
$\dot{\nu}_3$	0.0187	0.0006	0.1123	0.0411	**10.3880**	0.0133	0.0024	0.0843	0.0166	...
$\dot{\nu}_4$	0.0087	0.1908	0.0171	0.0023	0.000	0.0001	0.0520	0.0015	0.0016	...
$\dot{u}$	0.0008	0.0007	0.0078	0.0194	0.0167	0.0250	0.0054	0.0223	0.0107	...

3.6.4 A Stochastic Parameterized Extended Kalman Filter (SPEKF) Model: Incorporating Data Assimilation into the CEBoosting Algorithm

In many practical situations, the observations are only available for a subset of state variables, known as partial observations. It will be shown in the following that the CEBoosting

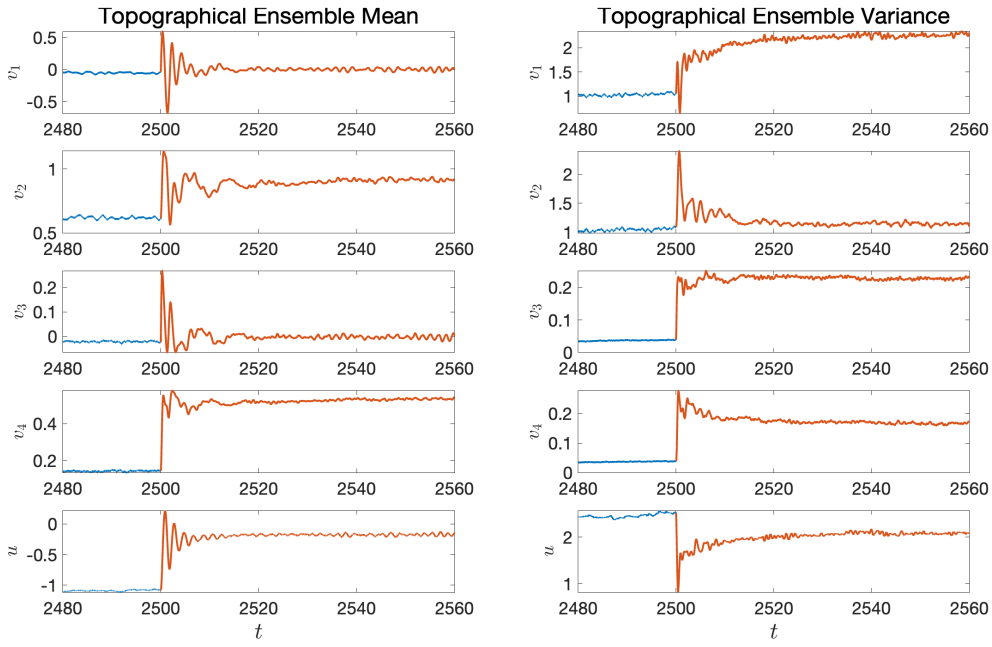

Figure 3.6: Ensemble mean and variance of each state variable in the topographic model. Regime switching happens at $t = 2500$.

algorithm can be naturally applied to the case with partial observations when data assimilation is appropriately incorporated. To illustrate the CEBoosting algorithm in the partial observational scenario, a simple yet practically useful nonlinear model is utilized as a testbed. The model is the so-called stochastic parameterized extended Kalman filter (SPEKF) model [265, 266],

$$\begin{aligned}
\frac{du}{dt} &= [-(\gamma+\hat{\gamma}) + i(\omega+\hat{\omega})]u + (b+\hat{b}) + \sigma_u \dot{W}_u, \\
\frac{d\gamma}{dt} &= -d_\gamma \gamma + \sigma_\gamma \dot{W}_\gamma, \\
\frac{d\omega}{dt} &= -d_\omega \omega + \sigma_\omega \dot{W}_\omega, \\
\frac{db}{dt} &= -d_b b + \sigma_b \dot{W}_b.
\end{aligned} \tag{3.17}$$

In this SPEKF model, $u(t)$ is a complex-valued state variable and is the only variable in the system to be observed. The observed variable $u(t)$ is driven by three hidden variables $\gamma(t), \omega(t)$ and $b(t)$. The parameters d_γ, d_ω, d_b are all positive, serving as damping factors.

The parameters σ_u, σ_γ, σ_ω and σ_b are noise coefficients, which are also positive. The white noises $\dot{W}_u$ and $\dot{W}_b$ are complex-valued while $\dot{W}_\gamma$ and $\dot{W}_\omega$ are real. The governing equations of $\gamma(t)$, $\omega(t)$ and $b(t)$ are Ornstein–Uhlenbeck (OU) processes [267] with γ and ω taking real values and b taking a complex value. The three constants $\hat{\gamma}$, $\hat{\omega}$ and $\hat{b}$ in the dynamics of u represent the mean damping, mean phase, and the mean forcing, respectively.

The SPEKF model (3.17) has been widely used as an approximate model to describe a spectral mode of a complex turbulent system, especially in the context of data assimilation and ensemble prediction [4, 64, 266]. Physically, the variable $u(t)$ represents one of the resolved modes (i.e., observable) in the turbulent signal, while the three hidden variables $\gamma(t), \omega(t)$ and $b(t)$ are surrogates for the nonlinear interaction between $u(t)$ and other unobserved modes in the original governing equation after applying the spectral decomposition. The idea of the SKEPF model is that the small or unresolved scale variables are stochastically parameterized by inexpensive linear and Gaussian processes, representing stochastic damping $\gamma(t)$, stochastic phase $\omega(t)$ and stochastic forcing $b(t)$. Despite the model error in using such Gaussian approximations for the original unresolved nonlinear dynamics, these Gaussian processes succeed in providing accurate statistical feedback from the unresolved scales to the resolved ones. Thus the intermittency and non-Gaussian features observed in the resolved variables can be accurately recovered. The statistics in the SPEKF model can also be solved with exact and analytic formulae, which allow an accurate and efficient estimation of the model states. The SPEKF type of model has been used for filtering multiscale turbulent dynamical systems [64], stochastic superresolution [268], and filtering Navier–Stokes equations with model error [269]. It has been shown that the SPEKF model has much higher skill than classical Kalman filters using the so-called mean stochastic model (MSM) to capture the irregularity and intermittency in nature.

In the following, the system will experience regime switching twice. In the starting regime (Regime 1), all three hidden variables have significant contributions to the dynamics of the observed process u. In Regime 2, ω will be removed from the dynamics of u; therefore, the stochastic phase does not influence the dynamics of the resolved variable u. Finally, in Regime 3, the stochastic damping γ will be removed from the dynamics of u, but the contribution from ω will be added back. It should be noted that the variables removed from the dynamics of u continue to evolve acorrding to their own governing equations, and simply no longer contribute to the evolution of u. Figure 3.7 shows the observed trajectories and the associated statistics of the real part of the observed u variable. Due to the stochastic

damping, the time series in Regime 1 and Regime 2 are intermittent with multiple extreme events when the overall damping $\hat{\gamma}+\gamma(t)$ becomes positive. As a result, the PDFs are non-Gaussian fat-tailed. In contrast, very few extreme events are found in Regime 3, and the associated PDF is Gaussian. Similarly, the ACF in Regime 2 shows a clear oscillatory pattern when it decays. This is due to a dominant frequency of the time series coming from the constant phase $\hat{\omega}$. Such a regular oscillation in the ACF becomes less significant when the stochasticity is added to the phase term via $\omega(t)$.

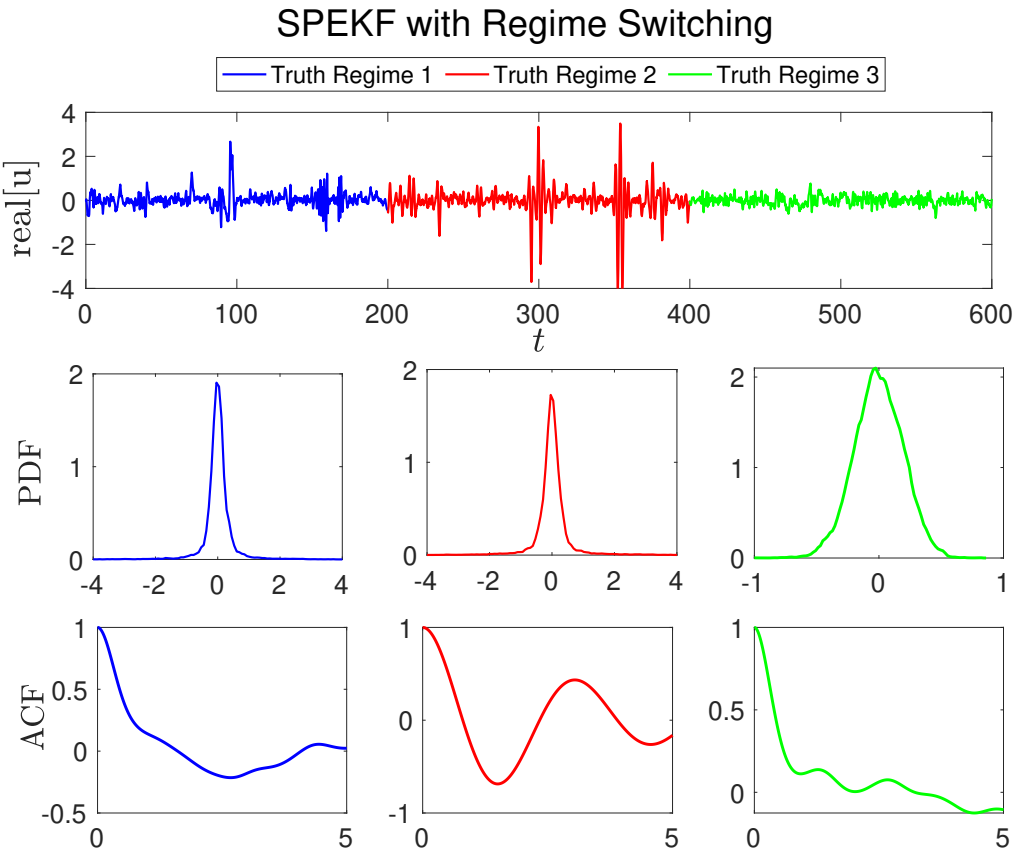

Figure 3.7: The SPEKF model with three different regimes (twice the regime switching) as time evolves. In Regime 1, all three hidden variables γ, ω, and b have significant contributions to the dynamics of u. In Regime 2, ω is removed from the dynamics of u. In Regime 3, the stochastic damping γ is removed from the dynamics of u and the contribution from ω is added back. First row: the time series of the real part of the observed variable u. Second row: the PDF of u in each regime. Third row: the ACF of u in each regime.

The goal here is to detect the regime switching and reveal the dynamics in each regime. It is worth highlighting that the dimensions of the system in the three regimes are different. In the absence of ω or γ, the dimension reduces from 4 to 3 from Regime 1 to Regimes 2 and 3. However, such a change is unknown in practice and relies on the learning algorithm to detect it. In particular, there is no observed time series of γ, ω, and b. Therefore, recovering these variables becomes an essential step in the online learning algorithm. To this end, the following procedure is adopted that incorporates data assimilation into the CEBoosting algorithm. Assume the model in Regime 1 is known. Each time when the new batch of time series of u is obtained, such a model is utilized to sample a trajectory of the three unobserved variables γ, ω, and b conditioned on the observed trajectory of u. The sampled trajectory can be thought of as the analog of one ensemble member of the ensemble Kalman smoother solution of the system conditioned on the observed signal of u [270], although the sampled trajectory using the SPEKF model can be written down using closed analytic formulae [194]. See [200] for the implementation details of using such closed analytic formulae for data assimilation. This augments the unobserved components of each batch of data. Then the CEBooting algorithm is utilized to compute the causal relationships in light of the observed time series of u and the sampled time series of γ, ω, and b. If the causation entropy from any function involving γ to the dynamics of u is zero, then the γ equation is eliminated from the final model structure. A similar logic applies to ω and b. Note that as u and b are complex-valued variables, the actual computation regards the real and imaginary parts as two processes in computing the causation entropy.

The main focus here is on identifying the dynamics of u. In particular, we aim to investigate if all three unobserved variables contribute to the observed process of u. To this end, the linear Gaussian models of γ, ω, and b are assumed to be fixed. A function library is designed with $\{u\gamma, u\omega, b, u^2, \gamma\omega, \gamma^2, \omega^2, b^2, \gamma b, \omega b, ub\}$ to learn the structure of u.

Figure 3.8 shows the real part of u with regime switching and sampled trajectory of γ, ω, b across the three regimes. When the unobserved variables γ, ω, or b contribute to the dynamics of u, the sampled processes match the truth quite well. On the other hand, when ω and γ disappear in Regimes 2 and 3, respectively, the sampling result provides random trajectories. The causation entropy from the terms involving these trajectories has no contribution to the dynamics of u. The first row of Table 3.8, showing the causation entropies of Regime 1, is based on a time series with 200 time units. The second and the third rows show the causation entropies for the detected Regime 2 and Regime 3, respectively,

using one batch of data with a length of 20 units. It is seen that the term $u\omega$ has a nearly zero causation entropy to $\dot{u}$ in Regime 2, where ω is sampled. Similarly, the sampled γ leads to a nearly zero causation entropy from $u\gamma$ to $\dot{u}$ in Regime 3.

Figure 3.9 shows the conditional mean and uncertainty (two standard deviations) of γ and ω from the smoother solution. It is seen that when γ and ω contribute to the dynamics of u in Regime 1, the conditional mean follows the truth quite well. However, the conditional mean of ω and γ behaves like a random trajectory in Regime 2 and Regime 3, respectively, with relatively large uncertainty. Thus, the sampled trajectories are formed from randomness and uncertainty, with no causal inference on the observed variable u.

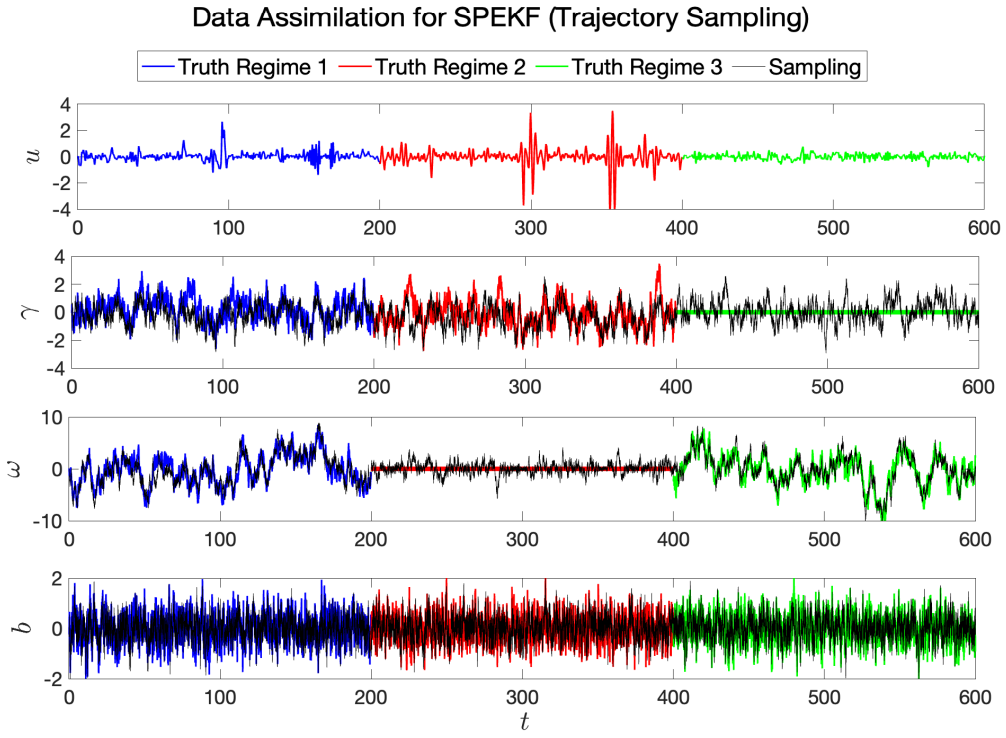

Figure 3.8: Trajectory sampling for the SPEKF model with twice the regime switching. The black curve represents the conditional sampling of each hidden process given the observed variable u in all three regimes.

Table 3.8: The SPEKF model - The CEM with Data Assimilation. First row: causation entropy with existing regime 1 data (200 time units). Second row: first batch (20 time units) causation entropy of regime 2. Third row: first batch (20 time units) causation entropy of regime 3.

	$u\gamma$	$u\omega$	b	u^2	$\gamma\omega$	γ^2	ω^2	b^2	γb	ωb	ub
$\dot{u}^{(1)}$	**0.2527**	**0.5272**	**0.2289**	0.0077	0.0006	0.0028	0.0003	0.0011	0.0090	0.0084	0.0137
$\dot{u}^{(2)}$	**0.4525**	0.0208	**0.1163**	0.0491	0.0070	0.0080	0.0019	0.0015	0.0113	0.0035	0.0066
$\dot{u}^{(3)}$	0.0844	**0.7032**	**0.3486**	0.0048	0.0027	0.0019	0.0017	0.0001	0.0068	0.0106	0.0063

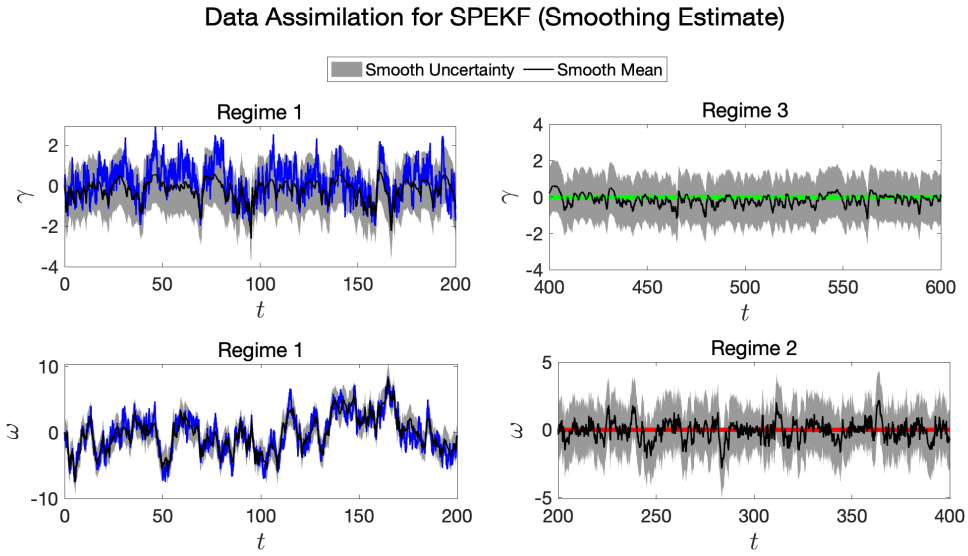

Figure 3.9: The smoother mean and the smoother uncertainty (two standard deviations) of γ and ω from the SPEKF model with twice the regime switching. The black curve shows the posterior mean of each hidden process conditioned on the observed u. The shaded gray area correspond to the 95% confidence interval (two standard deviations) of the corresponding mean estimate.

3.7 Conclusion

Online nonlinear system identification with sequential data has recently become important in many applications, e.g., extreme weather events, climate change, and autonomous systems. In this work, we developed a causation entropy boosting (CEBoosting) framework for online nonlinear system identification. The CEBoosting algorithm aims to (i) discover a sparse residual model structure based on the aggregated causation entropy calculated from sequential data and (ii) calibrate the residual model with the identified sparse structure via least square estimation. If the true system experiences multiple regime switching, the proposed framework gradually identifies a summation of residual models, which has a close analogy to the statistical technique of boosting. We tested the proposed framework for complex systems with features including chaotic behavior, high dimensionality, intermittency and extreme events, and partial observations. The results show that the CEBoosting method can capture the regime switching and then calibrate residual models for various types of complex dynamical based on a limited amount of sequential data. It is worth noting that constraints can be naturally added to the learning algorithm. One important constraint is the so-called physics constraint [191], which requires the total energy in the quadratic nonlinear terms to be conserved. It guarantees the long-term stability of the identified system. Such a constraint has yet to be explicitly incorporated into the current framework, although the resulting parameters in various non-Gaussian test cases shown in this work already roughly satisfy this constraint. Adding constraints can be easily achieved by imposing simple relationships between model parameters in the parameter estimation step, which still allows using closed analytic formulae for finding the parameters. See, for example, [200] for details. Other future work includes the uncertainty quantification of the aggregated causation entropy and the further study of other causality metrics.

Chapter 4

Modeling Partially Observed Complex Dynamical Systems and Efficient Data Assimilation

A new knowledge-based and machine learning hybrid modeling approach, called conditional Gaussian neural stochastic differential equation (CGNSDE), is developed to facilitate modeling complex dynamical systems and implementing analytic formulae of the associated data assimilation (DA). In contrast to the standard neural network predictive models, the CGNSDE is designed to effectively tackle both forward prediction tasks and inverse state estimation problems. The CGNSDE starts by exploiting a systematic causal inference via information theory to build a simple knowledge-based nonlinear model that nevertheless captures as much explainable physics as possible. Then, neural networks are supplemented to the knowledge-based model in a specific way, which not only characterizes the remaining features that are challenging to model with simple forms but also advances the use of analytic formulae to efficiently compute the nonlinear DA solution. These analytic formulae are used as an additional computationally affordable loss to train the neural networks that directly improve the DA accuracy. This DA loss function promotes the CGNSDE to capture the interactions between state variables and thus advances its modeling skills. With the DA loss, the CGNSDE is more capable of estimating extreme events and quantifying the associated uncertainty. Furthermore, crucial physical properties in many complex systems, such as the translate-invariant local dependence of state variables, can significantly simplify

the neural network structures and facilitate the CGNSDE to be applied to high-dimensional systems. Numerical experiments based on chaotic systems with intermittency and strong non-Gaussian features indicate that the CGNSDE outperforms knowledge-based regression models, and the DA loss further enhances the modeling skills of the CGNSDE.

Deep learning is widely used to predict complex dynamical systems in many scientific and engineering areas. However, the black-box nature of these deep learning models presents significant challenges for carrying out simultaneous data assimilation (DA), which is a crucial technique for state estimation, model identification, and reconstructing missing data. Integrating ensemble-based DA methods with nonlinear deep learning models is computationally expensive and may suffer from large sampling errors. To address these challenges, we introduce a deep learning framework designed to simultaneously provide accurate forecasts and efficient DA. It is named Conditional Gaussian Koopman Network (CGKN), which transforms general nonlinear systems into nonlinear neural differential equations with conditional Gaussian structures. CGKN aims to retain essential nonlinear components while applying systematic and minimal simplifications to facilitate the development of analytic formulae for nonlinear DA. This allows for seamless integration of DA performance into the deep learning training process, eliminating the need for empirical tuning as required in ensemble methods. CGKN compensates for structural simplifications by lifting the dimension of the system, which is motivated by Koopman theory. Nevertheless, CGKN exploits special nonlinear dynamics within the lifted space. This enables the model to capture extreme events and strong non-Gaussian features in joint and marginal distributions with appropriate uncertainty quantification. We demonstrate the effectiveness of CGKN for both prediction and DA on three strongly nonlinear and non-Gaussian turbulent systems: the projected stochastic Burgers–Sivashinsky equation, the Lorenz 96 system, and the El Niño-Southern Oscillation. The results justify the robustness and computational efficiency of CGKN.

A discrete-time conditional Gaussian Koopman network (CGKN) is developed in this work to learn surrogate models that can perform efficient state forecast and data assimilation (DA) for high-dimensional complex dynamical systems, e.g., systems governed by nonlinear partial differential equations (PDEs). Focusing on nonlinear partially observed systems that are common in many engineering and earth science applications, this work exploits Koopman embedding to discover a proper latent representation of the unobserved system states, such that the dynamics of the latent states are conditional linear, i.e., linear with the

given observed system states. The modeled system of the observed and latent states then becomes a conditional Gaussian system, for which the posterior distribution of the latent states is Gaussian and can be efficiently evaluated via analytical formulae. The analytical formulae of DA facilitate the incorporation of DA performance into the learning process of the modeled system, which leads to a framework that unifies scientific machine learning (SciML) and data assimilation. The performance of discrete-time CGKN is demonstrated on several canonical problems governed by nonlinear PDEs with intermittency and turbulent features, including the viscous Burgers' equation, the Kuramoto–Sivashinsky equation, and the 2-D Navier–Stokes equations, with which we show that the discrete-time CGKN framework achieves comparable performance as the state-of-the-art SciML methods in state forecast and provides efficient and accurate DA results. The discrete-time CGKN framework also serves as an example to illustrate unifying the development of SciML models and their other outer-loop applications such as design optimization, inverse problems, and optimal control.

4.1 Conditional Gaussian Neural Stochastic Differential Equation

Complex dynamical systems appear in many areas, including geophysics, climate science, engineering, neural science, and material science [1–4, 102, 173, 271–274]. These systems are highly nonlinear and are often strongly chaotic or turbulent [7, 273, 275]. Intermittency, extreme events, and non-Gaussian probability density functions (PDFs) are some of the typical features in these systems [8–10, 12, 276]. Developing appropriate models to describe the observed features of these complex systems is crucial in exploring the underlying physics and simulating the associated phenomena. These models also play an essential role in estimating the states via data assimilation (DA) [64, 65, 96, 187, 277], which is the pre-requite for statistical forecast and the uncertainty quantification of the system across different spatial-temporal scales [11, 278–281].

Nonlinear ordinary or partial differential equations (ODEs or PDEs) were standard techniques to model these complex phenomena. To characterize the multiscale features and the uncertainty in nature, stochastic differential equations (SDEs) have also become popular modeling tools over the past few decades [282–285]. When applied to DA, these

differential equations act as the forecast models. Their short-term statistical prediction, known as the prior distribution, is combined with the available noisy partial observations via the Bayesian inference. The result is called the posterior distribution, which is the solution of DA. It is worth highlighting that approximations are typically incorporated into the development of practical models. These approximations are essential since the perfect knowledge of nature is seldom known. They are also crucial to reducing the computational cost that facilitates the analysis of the model properties and the implementation of DA and ensemble forecasts. Commonly used approaches include the development of physics-based reduced-order models (ROMs) with proper closures [17–19, 32, 33, 286, 287], stochastic parameterizations [66, 82–84], and data-driven surrogate models [42, 76–79, 195, 288, 289].

With the rapid growth of artificial intelligence in recent years, machine learning has become one of the dominant methods for studying complex dynamical systems [30, 88, 107, 145, 290–292], especially for DA and prediction [293–295]. On the one hand, machine learning has been widely used as a computationally efficient surrogate of the complicated knowledge-based forecast models [93, 296–300] or played the role as a statistical correction to the knowledge-based models that mitigates the model error in the forecast step of DA [87, 89, 90, 218, 259, 301–303]. It has also been exploited to optimize the tuning parameters in ensemble DA, such as the inflation rate of the covariance matrix [304–306]. On the other hand, machine learning has been applied more directly by building end-to-end learning schemes for the entire DA system [81, 94, 95, 307–309]. In terms of learning surrogate models with DA performed, an auto-differentiable DA tool [310] has been developed and derivative-free optimization technique [145] has been explored for the ensemble-based DA methods.

In this paper, a new hybrid knowledge-based and machine learning modeling approach, called conditional Gaussian neural stochastic differential equation (CGNSDE), is developed to facilitate modeling complex dynamical systems and implementing the associated DA. The CGNSDE starts by exploiting a systematic causal inference approach to build a simple knowledge-based nonlinear model that nevertheless captures as much explainable physics as possible. Then, neural networks are supplemented to the knowledge-based model, aiming to characterize the remaining features that are challenging to model with simple forms. Notably, the neural networks are combined with the knowledge-based model in a specific way that advances the use of analytic formulae to compute the nonlinear DA solution. In addition to significantly accelerating the computational efficiency when applying the CGNSDE to

online DA, these analytic formulae allow us to explicitly augment the standard loss function in training CGNSDE as the predictive model with an additional loss that evaluates the DA skill. The analytic formulae advance a rapid and effective way to incorporate both the path-wise error and the posterior uncertainty into the DA loss.

The CGNSDE has several unique features that distinguish it from many existing machine learning approaches in modeling complex systems and data assimilation. First, the explainable physical components are the critical building blocks of the CGNSDE. To this end, a computationally efficient causal inference via information theory is applied to systematically derive the knowledge-based nonlinear model. The robust model identification process aims to discover the explainable large-scale physics in multiscale complex turbulent systems. In contrast, the machine learning components in the hybrid model can be regarded as the statistical parameterizations of the small-scale or more complicated unresolved features of the underlying system. Second, both the knowledge-based nonlinear model and the entire hybrid model are required to satisfy the so-called conditional Gaussian nonlinear structures. It has been shown that the conditional Gaussian nonlinear structures are ubiquitous in describing or approximating many natural and engineering phenomena [193, 194, 311]. In addition to being physically consistent with nature, one salient feature of the conditional Gaussian nonlinear structure is that, despite the strong nonlinearity and non-Gaussian statistics, the conditional distribution of the unobserved states given the observations can be written down using closed analytic formulae. Such a conditional distribution is precisely the posterior distribution in DA. As a result, the analytically solvable statistics prevent the use of ensemble methods in DA, enhancing computational efficiency and avoiding empirical tuning to mitigate numerical sampling errors. Third, the efficiency in solving the conditional statistics analytically allows the incorporation of the DA loss into the loss function for training the neural network component, which naturally improves the skill of the CGNSDE in state estimation. Reciprocally, as the DA performance relies on the interdependence between different state variables, such an additional loss will advance the neural network to improve the identification of the causal relationship of the underlying system, further enhancing the modeling skills of the CGNSDE. With the explicit DA loss, the CGNSDE is more capable of estimating extreme events and quantifying the associated uncertainty.

4.1.1 Preliminaries on Conditional Gaussian Nonlinear System

The conditional Gaussian nonlinear system (CGNS) is a class of nonlinear and non-Gaussian SDEs, which has wide applications in various disciplines. The general expression of the CGNS is [193, 312]:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{u}_1}{\mathrm{d}t} &= \mathbf{f}_1(\mathbf{u}_1) + \mathbf{g}_1(\mathbf{u}_1)\mathbf{u}_2 + \boldsymbol{\sigma}_1(\mathbf{u}_1)\dot{\mathbf{W}}_1, \\ \frac{\mathrm{d}\mathbf{u}_2}{\mathrm{d}t} &= \mathbf{f}_2(\mathbf{u}_1) + \mathbf{g}_2(\mathbf{u}_1)\mathbf{u}_2 + \boldsymbol{\sigma}_2(\mathbf{u}_1)\dot{\mathbf{W}}_2, \end{aligned} \tag{4.1}$$

where $\mathbf{u}_1 \in \mathbb{R}^{N_1}$ and $\mathbf{u}_2 \in \mathbb{R}^{N_2}$ are the multi-dimensional state variables considered here. The vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ have the same dimension as $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively, while $\mathbf{g}_1 \in \mathbb{R}^{N_1 \times N_2}$ and $\mathbf{g}_2 \in \mathbb{R}^{N_2 \times N_2}$ are two matrices. The two vectors $\dot{\mathbf{W}}_1 \in \mathbb{R}^{N_1'}$ and $\dot{\mathbf{W}}_2 \in \mathbb{R}^{N_2'}$ are white noises, and the noise coefficients $\boldsymbol{\sigma}_1 \in \mathbb{R}^{N_1 \times N_1'}$ and $\boldsymbol{\sigma}_2 \in \mathbb{R}^{N_2 \times N_2'}$ are two matrices. The dimension N_1' of the noise vector $\dot{\mathbf{W}}_1$ does not necessarily equal that of the state variable $\mathbf{u}_1$ (similar for the dimensions of $\mathbf{u}_2$ and $\dot{\mathbf{W}}_2$), though in many applications they are set to be the same for simplicity. All the six vectors and matrices $\mathbf{f}_i$, $\mathbf{g}_i$ and $\boldsymbol{\sigma}_i$, with $i = 1, 2$, on the right-hand side of (4.1) can be any nonlinear functions of $\mathbf{u}_1$ and time t, though for notation simplicity the explicit time dependence is omitted. Due to such nonlinear dependence on $\mathbf{u}_1$, the system (4.1) is highly nonlinear in terms of the coupled state variables $(\mathbf{u}_1, \mathbf{u}_2)^T$. As a result, the marginal distributions $p(\mathbf{u}_1)$ and $p(\mathbf{u}_2)$ and joint distribution $p(\mathbf{u}_1, \mathbf{u}_2)$ can all be highly non-Gaussian. Nevertheless, both equations in (4.1) depend on $\mathbf{u}_2$ in a conditionally linear way. Therefore, given a trajectory of $\mathbf{u}_1$, the equations in (4.1) become a conditional linear system with respect to $\mathbf{u}_2$, and the conditional distribution $p(\mathbf{u}_2(t)|\mathbf{u}_1(s), s \leqslant t) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{R})$ is Gaussian. Such a conditional distribution is the posterior distribution of DA (more precisely the filtering solution), where the trajectory of $\mathbf{u}_1$ up to time t, namely $\mathbf{u}_1(s \leqslant t)$, is the observations while the state of $\mathbf{u}_2$ at t needs to be estimated. Notably, the conditional mean $\boldsymbol{\mu}$ and conditional covariance $\mathbf{R}$ can be solved by the closed analytic formulae:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{\mu}}{\mathrm{d}t} &= (\mathbf{f}_2 + \mathbf{g}_2\boldsymbol{\mu}) + (\mathbf{R}\mathbf{g}_1^T)(\boldsymbol{\sigma}_1\boldsymbol{\sigma}_1^T)^{-1}\left(\frac{\mathrm{d}\mathbf{u}_1}{\mathrm{d}t} - (\mathbf{f}_1 + \mathbf{g}_1\boldsymbol{\mu})\right), \\ \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t} &= \mathbf{g}_2\mathbf{R} + \mathbf{R}\mathbf{g}_2^T + \boldsymbol{\sigma}_2\boldsymbol{\sigma}_2^T - \mathbf{R}\mathbf{g}_1^T(\boldsymbol{\sigma}_1\boldsymbol{\sigma}_1^T)^{-1}(\mathbf{g}_1\mathbf{R}). \end{aligned} \tag{4.2}$$

Many complex nonlinear dynamical systems fit into the modeling framework (4.1). Some well-known classes of the models are physics-constrained nonlinear stochastic models

(for example the noisy versions of Lorenz models, low-order models of Charney-DeVore flows, and a paradigm model for topographic mean flow interaction), stochastically coupled reaction-diffusion models in neuroscience and ecology (for example stochastically coupled FitzHugh-Nagumo models and stochastically coupled SIR epidemic models), and multiscale models for geophysical flows (for example the Boussinesq equations with noise and stochastically forced rotating shallow water equation) [193]. This modeling framework has been exploited to develop realistic systems for the Madden-Julian oscillation and Arctic sea ice [313, 314].

In addition to modeling many natural phenomena, the CGNS framework and its closed analytic DA formulae have been applied to study many theoretical and practical problems. The framework has been utilized to develop a nonlinear Lagrangian DA algorithm, allowing rigorous analysis to study model error and uncertainty [315–317]. The analytically solvable DA scheme has been applied to the state estimation and the prediction of intermittent time series for the monsoon and other climate phenomena [318, 319]. Notably, the efficient DA procedure also helps develop a rapid algorithm to solve high-dimensional Fokker-Planck equation [70, 257]. The classical Kalman-Bucy filter [63] is the simplest special example of (4.2).

It is also worth highlighting that the ideas of the CGNS modeling framework and the associated DA procedure have been applied to a much wider range of problems. Examples include developing forecast models in dynamic stochastic superresolution [268, 320], building stochastic superparameterizations for geophysical turbulence [321–323], and designing efficient multiscale DA schemes [324, 325]. All these facts indicate that the CGNS provides a valuable building block for many practical methods.

When characterizing multiscale systems, the CGNS is quite effective in modeling large-scale features. Note that the nonlinearity in many applications, especially those in geophysics and fluids, is quadratic, which comes from advection or convection. If $\mathbf{u}_1$ and $\mathbf{u}_2$ are utilized to describe the large- and small-scale state variables, respectively, then the quadratic nonlinearities between $\mathbf{u}_1$ and itself and between $\mathbf{u}_1$ and $\mathbf{u}_2$ can be accurately captured. In some systems, cubic damping appears. CGNS can also involve such a strong damping in the large-scale dynamics. The major nonlinearity that the CGNS cannot fully characterize is the quadratic self-interactions among the small-scale variables, as $\mathbf{u}_2$ is only allowed to appear in a conditional linear way. In such a situation, stochastic parameterizations are often adopted to approximate the self-nonlinear interactions of $\mathbf{u}_2$. A straightforward approach is

to replace the quadratic terms of $\mathbf{u}_2$ by stochastic noise and additional damping [326, 327], which works well if $\mathbf{u}_2$ lies in the fast time scale. Yet, designing suitable approximate strategies with parsimonious and explicit expressions is generally a nontrivial task. This opens the doors for machine learning to supplement the CGNS.

4.1.2 CGNSDE for State Forecast and Data Assimilation

The CGNSDE is a hybrid knowledge-based and neural network version of the CGNS. It is required to have a similar structure as the CGNS in (4.1), but the functions that depend on $\mathbf{u}_1$ are generalized to involve neural networks. The CGNSDE reads:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{u}_1}{\mathrm{d}t} &= \widetilde{\mathbf{f}}_1(\mathbf{u}_1) + \widetilde{\mathbf{g}}_1(\mathbf{u}_1)\mathbf{u}_2 + \boldsymbol{\sigma}_1(\mathbf{u}_1)\dot{\mathbf{W}}_1, \\ \frac{\mathrm{d}\mathbf{u}_2}{\mathrm{d}t} &= \widetilde{\mathbf{f}}_2(\mathbf{u}_1) + \widetilde{\mathbf{g}}_2(\mathbf{u}_1)\mathbf{u}_2 + \boldsymbol{\sigma}_2(\mathbf{u}_1)\dot{\mathbf{W}}_2, \end{aligned} \tag{4.3}$$

where the functions with the terms with tildes represent the combinations of knowledge-based terms and neural networks that only depend on $\mathbf{u}_1$. More specifically, the CGNSDE in (4.3) can be rewritten into the following more detailed form:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{u}_1}{\mathrm{d}t} &= \mathbf{f}_1(\mathbf{u}_1) + \mathbf{g}_1(\mathbf{u}_1)\mathbf{u}_2 + \mathbf{NN}_{1,1}(\mathbf{u}_1) + \mathbf{NN}_{1,2}(\mathbf{u}_1)\mathbf{u}_2 + \boldsymbol{\sigma}_1(\mathbf{u}_1)\dot{\mathbf{W}}_1, \\ \frac{\mathrm{d}\mathbf{u}_2}{\mathrm{d}t} &= \mathbf{f}_2(\mathbf{u}_1) + \mathbf{g}_2(\mathbf{u}_1)\mathbf{u}_2 + \mathbf{NN}_{2,1}(\mathbf{u}_1) + \mathbf{NN}_{2,2}(\mathbf{u}_1)\mathbf{u}_2 + \boldsymbol{\sigma}_2(\mathbf{u}_1)\dot{\mathbf{W}}_2, \end{aligned} \tag{4.4}$$

where $\mathbf{f}_1, \mathbf{g}_1, \mathbf{f}_2$ and $\mathbf{g}_2$ are nonlinear functions of $\mathbf{u}_1$ with explainable forms (e.g., a nonlinear combination of some candidate functions), while $\mathbf{NN}_{i,j}$ with $i, j = 1, 2$ are neural network functions to further enhance the ability of the CGNSDE in capturing the variability of nature. The neural network components $\mathbf{NN}_{i,j}$, can be formalized using four distinct parts derived either from the output of a single neural network or from several neural networks. Note that the diffusion coefficients $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ in the CGNSDE can, in principle, be generalized as a combination of explainable functions and neural networks as well. Nevertheless, they are modeled by only the parametric forms in this work for simplicity. One key feature of (4.4) is that those neural network components only allow $\mathbf{u}_1$ as the input, which guarantees the conditional Gaussianity of the entire system. It is worth noting that the input of the neural network part does not necessarily have to be the value of $\mathbf{u}_1$ at the current time

t. It can consist of a segment of the trajectory of $\{\mathbf{u}_1(s)|s \leqslant t\}$. Input with such non-Markovian terms nevertheless preserves the conditional Gaussianity of the system. This feature allows a significant degree of freedom to the design of neural networks, allowing both the feed-forward and recurrent neural networks to apply to the CGNSDE. With the dependence of $\mathbf{f}_1, \mathbf{g}_1, \mathbf{f}_2$ and $\mathbf{g}_2$ only on $\mathbf{u}_1$ in the CGNSDE, the conditional distribution $p(\mathbf{u}_2(t)|\mathbf{u}_1(s), s \leqslant t) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{R})$, which is the posterior distribution of DA, is Gaussian and its time evolution can be written down using analytic formulae that are analogs to (4.2).

The CGNSDE has several unique advantages. First, the explainable physical components are the critical building blocks of the CGNSDE. This distinguishes the CGNSDE from the non-parametric models that replace the entire right-hand side of the original system with neural networks. These are helpful predictive models. However, these models are hard to be used to reveal the underlying physics. Such pure neural-network-based models are also challenging to effectively use for solving inverse problems, such as DA. With the large-scale features captured by explainable physical components, the neural networks play a more critical role in characterizing the residual. Therefore, they share many essential features as the residual networks that advance the performance of the neural networks in more efficiently capturing the features of the underlying dynamics [37, 328]. Second, the analytically solvable posterior distribution prevents the use of ensemble methods in DA. These analytic formulae not only enhance computational efficiency but also avoid empirical tunings in the standard ensemble DA that is essential to mitigate numerical sampling errors. Third, the efficient DA solver with these analytic formulae allows the incorporation of the DA loss into the target loss function to train the CGNSDE, which naturally improves the skill for state estimation. Reciprocally, as the performance of DA depends on the accuracy in modeling the interdependence between different state variables, namely their causal relationship, incorporating the DA loss will further enhance the performance of the CGNSDE in modeling the underlying dynamical features. With the explicit DA loss, the CGNSDE is also more capable of estimating extreme events and quantifying the associated uncertainty.

4.1.3 Procedure of Developing a CGNSDE

Assume that a sufficiently long time series of both $\mathbf{u}_1$ and $\mathbf{u}_2$ are available in the model development stage to determine the CGNSDE. In the testing stage for the online DA and forecast, only the observations of $\mathbf{u}_1$ are needed, which is consistent with the realistic

situations with partial observations.

The development of the CGNSDE involves two steps. First, the explicit expressions of the terms $\mathbf{f}_1(\mathbf{u}_1)+\mathbf{g}_1(\mathbf{u}_1)\mathbf{u}_2$ and $\mathbf{f}_2(\mathbf{u}_1)+\mathbf{g}_2(\mathbf{u}_1)\mathbf{u}_2$ need to be determined to include as much physically explainable information as possible. Second, suitable architectures and loss functions are designed for the neural networks $\mathbf{NN}_{i,j}$ with $i,j=1,2$. Notably, the order of the two steps is essential to prevent the neural networks from taking away the information described by the physically explainable components.

4.1.3.1 Determining the Knowledge-Based Components by Causal Inference

The knowledge-based components in the CGNSDE contain the terms $\mathbf{f}_1(\mathbf{u}_1)+\mathbf{g}_1(\mathbf{u}_1)\mathbf{u}_2$ and $\mathbf{f}_2(\mathbf{u}_1)+\mathbf{g}_2(\mathbf{u}_1)\mathbf{u}_2$ in (4.4). These functions are often determined by human knowledge case by case. Yet, in a more general situation, an automatic learning algorithm is preferred to determine these terms systematically. To this end, a causal inference method based on the so-called causation entropy is adopted to achieve such a goal [20, 21, 202, 204, 221]. Causation entropy is an efficient machine-learning method for the sparse identification of dynamical systems. The causal relationship facilitates the discovery of the underlying explainable knowledge-based components of the system. Notably, the model identification results using the causation entropy method are more robust to noise or chaotic features than the standard least absolute shrinkage and selection operator (LASSO) regressions. It has been shown that in the presence of even slight random noise, both the covariate selection accuracy and the fraction of zero entries may decrease significantly [220] when applying the standard LASSO regression.

The procedure of applying the causation entropy in determining the physically explainable components of the system is as follows.

Step 1. Determining the state variables. The state variables of the CGNSDE are pre-determined. To facilitate the presentation, these variables are included into an N-dimensional column vector:

$$\mathbf{U}=(\mathbf{u}_1,\mathbf{u}_2)^{\mathsf{T}}=(u_1,\ldots,u_{N_1},u_{N_1+1},\ldots,u_{N_1+N_2})^{\mathsf{T}}, \tag{4.5}$$

where $N=N_1+N_2$.

Step 2. Developing a function library. After determining the state variables, a library $\mathbf{h}$

consisting of in total M possible candidate functions to describe the right-hand side of the model is developed,

$$\mathbf{h} = \{h_1, \ldots, h_{m-1}, h_m, h_{m+1}, \ldots, h_M\}. \tag{4.6}$$

Typically, a large number of candidate functions is included in the library to allow for coverage over different possible dynamical features of the underlying true dynamics. Each h_m is given by a linear or nonlinear function containing a few components of $\mathbf{U}$. To follow the structure of CGNS in (4.1), the components within $\mathbf{u}_2$ are required to be linear in the candidate function h_m. Prior knowledge of the dynamics can help determine the function library. The library can also include as many potential candidate functions as possible provided that the elements within $\mathbf{u}_2$ are incorporated linearly, allowing the causal inference to determine the useful ones automatically.

Step 3. Computing the causation entropy. Next, causal inference is utilized to determine the model structure. To this end, a causation entropy $C_{h_m \to \dot{u}_i | [\mathbf{h} \backslash h_m]}$ is computed to detect if the candidate function h_m contributes to the right-hand side of the equation for u_i, namely $du_i/dt := \dot{u}_i$. The causation entropy is given by [202, 204, 221]:

$$C_{h_m \to \dot{u}_i | [\mathbf{h} \backslash h_m]} = H(\dot{u}_i | [\mathbf{h} \backslash h_m]) - H(\dot{u}_i | \mathbf{h}), \tag{4.7}$$

where $\mathbf{h} \backslash h_m$ represent the set that contains all functions in $\mathbf{h}$ except h_m. In other words, $\mathbf{h} \backslash h_m$ contains $M - 1$ candidate functions and is defined as

$$\mathbf{h} \backslash h_m = \{h_1, \ldots, h_{m-1}, h_{m+1}, \ldots, h_M\}. \tag{4.8}$$

The term $H(\cdot|\cdot)$ is the conditional entropy, which is related to Shannon's entropy $H(\cdot)$ and the joint entropy $H(\cdot, \cdot)$. For two multi-dimensional random variables $\mathbf{X}$ and $\mathbf{Y}$ (with the corresponding states being $\mathbf{x}$ and $\mathbf{y}$), they are defined as [329]:

$$\begin{aligned} H(\mathbf{X}) &= -\int_{\chi} p(\mathbf{x}) \log(p(\mathbf{x})) d\mathbf{x}, \\ H(\mathbf{Y}|\mathbf{X}) &= -\int_{\mathbf{x}} \int_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) \log(p(\mathbf{y}|\mathbf{x})) d\mathbf{y} d\mathbf{x}, \\ H(\mathbf{X}, \mathbf{Y}) &= -\int_{\mathbf{x}} \int_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) \log(p(\mathbf{x}, \mathbf{y})) d\mathbf{y} d\mathbf{x}, \end{aligned} \tag{4.9}$$

where $p(\mathbf{x})$ is the PDF of $\mathbf{x}$ and $p(\mathbf{y}|\mathbf{x})$ is the conditional PDF of $\mathbf{y}$ on $\mathbf{x}$. On the right-hand side of (4.7), the difference between the two conditional entropies indicates the information in $\dot{u}_i$ contributed by the specific function h_m given the contributions from all the other functions. Thus, it tells if h_m provides additional information to $\dot{u}_i$ conditioned on the other potential terms in the dynamics. It is worthwhile to highlight that the causation entropy in (4.7) is fundamentally different from directly computing the correlation between $\dot{u}_i$ and h_m, as the causation entropy also considers the influence of the other library functions. If both $\dot{u}_i$ and h_m are caused by a common factor $h_{m'}$, then $\dot{u}_i$ and h_m can be highly correlated. Yet, in such a case, the causation entropy $C_{h_m \to \dot{u}_i | [\mathbf{h} \setminus h_m]}$ will be zero as h_m is not the causation of $\dot{u}_i$.

The causation entropy is computed from each of the candidate functions in $\mathbf{h}$ to each $\dot{u}_i$. Thus, there are in total $N \times M$ causation entropies, which can be written as a $N \times M$ matrix, called the causation entropy matrix. Note that the dimension $\mathbf{X}$ in (4.9) is M when it is applied to compute the second term on the right-hand side of the causation entropy in (4.7). This implies that the direct calculation of the entropies in (4.9) involves a high-dimensional numerical integration, which is a well-known computationally challenging issue [238]. To circumvent the direct numerical integration, the entropy calculation approximates all the joint and marginal distributions as Gaussian. In such a way, the causation entropy can be computed by

$$\begin{aligned} C_{\mathbf{Z} \to \mathbf{X}|\mathbf{Y}} &= H(\mathbf{X}|\mathbf{Y}) - H(\mathbf{X}|\mathbf{Y},\mathbf{Z}) \\ &= H(\mathbf{X},\mathbf{Y}) - H(\mathbf{Y}) - H(\mathbf{X},\mathbf{Y},\mathbf{Z}) + H(\mathbf{Y},\mathbf{Z}) \\ &\approx \frac{1}{2}\ln(\det(\mathbf{R}_{\mathbf{XY}})) - \frac{1}{2}\ln(\det(\mathbf{R}_{\mathbf{Y}})) - \frac{1}{2}\ln(\det(\mathbf{R}_{\mathbf{XYZ}})) + \frac{1}{2}\ln(\det(\mathbf{R}_{\mathbf{YZ}})), \end{aligned} \tag{4.10}$$

where $\mathbf{R}_{\mathbf{XYZ}}$ denotes the covariance matrix of the state variables $(\mathbf{X},\mathbf{Y},\mathbf{Z})$ and similar for other covariance. The notations $\ln(\cdot)$ and $\det(\cdot)$ are the logarithm of a number and the determinant of a matrix, respectively.

The simple and explicit expression in (4.10) based on the Gaussian approximation can efficiently compute the causation entropy. It allows the computation of the causation entropy with a moderately large dimension, sufficient for many applications. It is worth noting that the Gaussian approximation may lead to certain errors in computing the causation entropy if the true distribution is highly non-Gaussian. Nevertheless, the primary goal is not to obtain the exact value of the causation entropy. Instead, it suffices to detect if the

causation entropy $C_{h_m \to \dot{u}_i | [\mathbf{h} \setminus h_m]}$ is nonzero (or practically above a small threshold value). In most applications, if a significant causal relationship is detected in the higher-order moments, it is very likely in the Gaussian approximation. This allows us to efficiently determine the sparse model structure, where the exact values of the nonzero coefficients on the right-hand side of the model will be calculated via a simple maximum likelihood estimation to be discussed in the following. Note that the Gaussian approximation has been widely applied to compute various information measurements and leads to reasonably accurate results [11, 188, 239–241].

With the $N \times M$ causation entropy matrix in hand, the next step is determining the model structure. This can be done by setting up a threshold value of the causation entropy and retaining only those candidate functions with the causation entropies exceeding the threshold. This will exclude those terms with small values of causation entropy, which is usually due to numerical error using a finite time series. The resulting model will contain only functions that significantly contribute to the dynamics and facilitate a sparse model structure. Sparsity is crucial to discovering the correct underlying physics and prevents overfitting [105, 330]. It will also guarantee the robustness of the model in response to perturbations and allow the model to apply to certain extrapolation tests.

Step 4. Parameter estimation. The final step is to estimate the parameters in the resulting model. Despite the nonlinearity in the underlying dynamics, the parameters usually appear in a linear way. In such a case, parameter estimation can be easily handled using a simple regression method or a maximum likelihood estimator. See [200] for the technical details. Notably, closed analytic formulae are available, making the procedure efficient and accurate. It is worth highlighting that constraints to the parameter values are often included in the parameter estimation procedure. In many turbulent systems, the quadratic nonlinear terms are assumed to be energy conserved. This is motivated by most geophysical systems where the quadratic nonlinearity is the advection and is a natural conservation quantity. The energy-conserving quadratic nonlinearity prevents the finite time blowup of the solution in the derived model and is physically consistent [191, 192]. Remarkably, closed analytic formulae are still available for parameter estimation in the presence of such constraints.

4.1.3.2 Determining the Neural Network Components

Although Step 4 in Section 4.1.3.1 provides a direct way to estimate the parameters in the knowledge-based components $\mathbf{f}_1(\mathbf{u}_1), \mathbf{g}_1(\mathbf{u}_1)\mathbf{u}_2, \mathbf{f}_2(\mathbf{u}_1), \mathbf{g}_2(\mathbf{u}_1)\mathbf{u}_2$, the parameters of these physically explainable terms in the CGNSDE are estimated in a slightly different way. Specifically, these parameters are not immediately estimated after the model format is determined by the causal inference. Once the model structure of the physically explainable components is determined (Steps 1-3 in Section 4.1.3.1), the neural network and the parameters of the physically explainable components are determined by a joint learning algorithm. This allows a simultaneous optimization of the two components in the CGNSDE.

It is worth noting that the CGNSDE in (4.4) is implemented to support automatic differentiation, which facilitates the training process using gradient descent methods. Training the neural network components in the CGNSDE involves using a standard forecast loss and a DA loss. The latter plays a vital role in improving the skillful DA using the CGNSDE. Notably, the closed analytic formulae of the DA solution facilitate the incorporation of such an additional crucial loss function.

The short-term forecast loss

Denote by $\tilde{\mathbf{U}}(t_n) = (\tilde{\mathbf{u}}_1(t_n), \tilde{\mathbf{u}}_2(t_n))^\mathsf{T}$, for $n = 1, \ldots, N_s$, which is a multi-dimensional series predicted by the CGNSDE (4.4) starting from $\mathbf{U}(t_0)$. The hyper-parameter N_s, which determines the forecast horizon, is usually a relatively small integer since the path-wise forecast diverges quickly for chaotic systems.

The forecast loss L_{forecast} is defined as the averaged error between the predicted states $\{\tilde{\mathbf{U}}(t_n)\}_{n=1}^{N_s}$ and the truth $\{\mathbf{U}(t_n)\}_{n=1}^{N_s}$:

$$L_{\text{forecast}}(\mathbf{U}, \tilde{\mathbf{U}}) := \frac{1}{N_s} \sum_{n=1}^{N_s} \|\mathbf{U}(t_n) - \tilde{\mathbf{U}}(t_n)\|^2, \tag{4.11}$$

where $\|\cdot\|$ is the standard vector ℓ^2-norm.

In each training epoch, a time series of $\mathbf{U}$ with N_s time steps is randomly sampled from the training data as the truth. Starting from the same initial value as the truth, the CGNSDE will be integrated for N_s steps to obtain the predicted states. Utilizing an auto-differentiation engine, the neural networks can be trained using the stochastic gradient descent (SGD) algorithm. In this work, an epoch is defined as a single iteration in which SGD is executed

using a selected batch.

The DA loss

The DA loss assesses the error in the DA solution computed from the CGNSDE. Notably, the explicit formulae in (4.2) are used to facilitate the calculation of the DA loss.

One pre-requite of carrying out the DA is to determine the noise coefficients $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ in (4.4). For the simplicity of presenting the idea, hereafter the dimensions of the white noise $\dot{\mathbf{W}}_1$ and $\dot{\mathbf{W}}_2$ in (4.4) are assumed to be the same as the state $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively, e.g., $N_1' = N_1$ and $N_2' = N_2$. Further assume $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ are diagonal positive definite matrix. With a pre-trained CGNSDE by only using the forecast loss, the diagonal elements in $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ can be estimated by the quadratic variation:

$$\mathrm{diag}(\boldsymbol{\sigma}_i) = \sqrt{\frac{\Delta t}{N_t}\sum_{n=1}^{N_t}\big(\dot{\mathbf{u}}_i(t_n) - \tilde{\dot{\mathbf{u}}}_i(t_n)\big)\odot\big(\dot{\mathbf{u}}_i(t_n) - \tilde{\dot{\mathbf{u}}}_i(t_n)\big)}, \qquad \text{for } i = 1,2, \tag{4.12}$$

where Δt is the numerical integration time step, N_t is the total number of the time steps in the training data, and the notation $\odot$ denotes the element-wise product. In (4.12), the time derivative of the true signal, namely $\dot{\mathbf{u}}_i(t_n)$, can often be obtained by calculating the numerical derivatives of the true state $\mathbf{u}_i$ at time t_n, while $\tilde{\dot{\mathbf{u}}}_i(t_n)$ is the time derivative of the corresponding prediction from the pre-trained CGNSDE. With the estimated $\boldsymbol{\sigma}_i$, the posterior mean $\boldsymbol{\mu}$ and covariance $\mathbf{R}$ of the unobserved variables $\mathbf{u}_2$ in CGNSDE can be calculated from the closed analytic formulae in (4.2), given the time series of the observed variables $\mathbf{u}_1$.

One natural way of constructing a DA loss L_{DA} is to compute the difference between the posterior mean estimate $\boldsymbol{\mu}$ and the true signal $\mathbf{u}_2$:

$$L_{DA}(\mathbf{u}_2, \boldsymbol{\mu}) := \frac{1}{N_l - N_b}\sum_{n=N_b+1}^{N_l}\|\mathbf{u}_2(t_n) - \boldsymbol{\mu}(t_n)\|^2, \tag{4.13}$$

where N_l denotes the number of steps to compute the DA solution during the training period. Note that N_l is usually much larger than N_s in (4.11). This is because the DA solution will not have a quick diverge as the forecast one. Note that although it is natural to set the forecast initial value to be the same as the truth to eliminate the initial inconsistency in the forecast loss, the exact initial distribution of the DA solution (especially the initial

uncertainty) is usually unknown. Therefore, it takes some time for the DA solution to be adjusted to eliminate the inconsistency from the initialization. To this end, the few steps of DA, which correspond to the burn-in period, are omitted, which leads to the DA loss being computed from the (N_b+1)-th step.

It is worth highlighting that the DA loss in (4.13) has a straightforward form and can be easily applied in practice. Although only the posterior mean is explicitly adopted in the loss function, the entire posterior information from DA is exploited since the posterior mean depends on the posterior covariance, as can be seen in (4.2). The DA loss in (4.13) will be adopted in all the numerical experiments in Section 4.1.4. The results there will demonstrate that the advantage of including this DA loss in training the CGNSDE to improve the modeling and DA skills, including performing more stable long-term simulations and better capturing the true system behaviors (e.g., the critical statistical properties, the chaotic patterns, and the extreme events).

An alternative DA loss

One potential shortcoming of the path-wise DA loss in (4.13) is that it does not fully use the probabilistic features in the state estimation via DA. Therefore, an alternative DA loss L_{DA} can be defined from a more probabilistic perspective based on the log-likelihood:

$$\begin{aligned} L_{DA}(\mathbf{u}_2, \boldsymbol{\mu}, \mathbf{R}) &:= -\ln\Big(\prod_{n=N_b+1}^{N_l} p\big(\mathbf{u}_2(t_n)|\mathbf{u}_1(s), s \leqslant t_n\big)\Big) \\ &= \frac{1}{2} \sum_{n=N_b+1}^{N_l} \Big(N_2 \log(2\pi) + \ln(\det(\mathbf{R}(t_n))) + \|\mathbf{u}_2(t_n) - \boldsymbol{\mu}(t_n)\|^2_{\mathbf{R}(t_n)} \Big), \end{aligned} \tag{4.14}$$

where $\|\cdot\|^2_{\mathbf{R}}$ denotes a weighted ℓ^2-norm, i.e., $\|\boldsymbol{\mu}\|^2_{\mathbf{R}} = \boldsymbol{\mu}^{\mathsf{T}}\mathbf{R}^{-1}\boldsymbol{\mu}$. The second line in (4.14) utilizes the fact that $p(\mathbf{u}_2(t_n)|\mathbf{u}_1(s), s \leqslant t_n)$ follows a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}(t_n), \mathbf{R}(t_n))$, whose mean $\boldsymbol{\mu}(t_n)$ and covariance matrix $\mathbf{R}(t_n)$ can be obtained by simulating (4.2). Considering that $N_2 \log(2\pi)$ is just a constant factor for any given dynamical system, the main difference between (4.14) and (4.13) is that the DA loss based on log-likelihood leads to a weighted ℓ^2-norm, which allows using the covariance matrix $\mathbf{R}$ to normalizes the mismatch between $\mathbf{u}_2$ and $\boldsymbol{\mu}$. Building a loss function in such a way potentially enhances the importance of the posterior uncertainty in training the CGNSDE. Consequently, the DA skill in recovering intermittency and extreme events is expected to be improved with such a loss

function.

Training the CGNSDE with both forecast and DA losses

Given the expression of the short-term forecast loss (4.11) and the DA loss (4.13), the overall target loss function can be defined as a weighted sum of these two functions:

$$L_{\text{total}} := \lambda_1 \underbrace{\frac{1}{N_s}\sum_{n=1}^{N_s} \|\mathbf{U}(t_n) - \tilde{\mathbf{U}}(t_n)\|^2}_{L_{\text{forecast}}} + \lambda_2 \underbrace{\frac{1}{N_l - N_b}\sum_{j=N_b+1}^{N_l} \|\mathbf{u}_2(t_j) - \boldsymbol{\mu}(t_j)\|^2}_{L_{\text{DA}}}, \tag{4.15}$$

where the two constants λ_1 and λ_2 are the weights of the forecast loss and DA loss, respectively. In the following numerical experiments, the two weights are assigned as $\lambda_1 = 1/N$, which is the dimension of all the state variables $\mathbf{U}$, and $\lambda_2 = 1/N_2$, which is the dimension of unobserved state variables $\mathbf{u}_2$. With such a choice, L_{forecast} and L_{DA} become the average of the mean squared errors between the truth and the predictions and the state estimation.

In each training epoch, a time series of $\mathbf{U}$ with N_s steps and another time series of $\mathbf{u}_2$ with N_l steps are sampled from the training data. The numerical simulation of $\tilde{\mathbf{U}}$ and $\boldsymbol{\mu}$ are performed based on the CGNSDE solvers that support auto-differentiation. These time series are used to evaluate the total loss defined in (4.15), with which the auto-differentiation provides the gradient information that can be employed to optimize the unknown parameters in the CGNSDE via gradient descent methods. An epoch refers to a single iteration where SGD is executed with a selected batch.

4.1.4 Numerical Experiments

In all the numerical experiments, the results from the following three models are compared to demonstrate the skillful performance of the CGNSDE. Without loss of generality, the feed-forward network is utilized in all the experiments. These simulations aim to show the necessity of supplementing the neural network components into the knowledge-based models and the crucial role of incorporating the DA loss into the overall loss function. The three models used below are:

1. The knowledge-based regression model. It corresponds to CGNSDE in (4.4) without the neural network parts. The unknown functions in this model are constructed based on a library of candidate functions and calibrated via a causation-entropy-based

system identification method. The detailed procedures are described in Section 4.1.3.1 from Step 1 to Step 4. Alternatively, the prior knowledge from the users can also be exploited to build such a model, in which case the prior knowledge replaces the causal inference for the model development.

2. The CGNSDE without DA loss. It corresponds to the CGNSDE model in the form of (4.4) combining knowledge-based and neural network components. However, only the forecast loss in (4.11) is utilized to train the neural networks. Developing this model involves first identifying the knowledge-based components (Steps 1 to 3 in Section 4.1.3.1), and then jointly training them with the neural network components using only a forecast loss.

3. The CGNSDE with the DA loss. It corresponds to the CGNSDE model in the form of (4.4) and is trained with the total loss function (4.15) that combines both forecast loss and DA loss. Developing this model involves first identifying the knowledge-based components (Steps 1 to 3 in Section 4.1.3.1), and then jointly training them with the neural network components using both forecast loss and DA loss.

The performance of all three models are examined utilizing the following validation metrics:

(a). Forecast MSE. It is the MSE (4.11) between the predicted state $\tilde{\mathbf{U}}$ and the truth $\mathbf{U}$.

(b). DA MSE. It is the DA MSE in (4.13) between the posterior mean estimate $\boldsymbol{\mu}$ and the unobserved true state $\mathbf{u}_2$.

(c). DA negative log-likelihood (Neg-Log-Likelihood). It is the DA negative log-likelihood in (4.14) calculated based on the true value of the unobserved state $\mathbf{u}_2$ related to the Gaussian distribution built upon the posterior mean $\boldsymbol{\mu}$ and the posterior covariance $\mathbf{R}$. Note that the DA loss in training the neural networks is still based on the error in the posterior mean related to the truth using (4.13). The negative log-likelihood (4.14) based on the entire posterior distribution is only used here as a validation criterion.

It is worth highlighting that the negative log-likelihood function utilizes the direction information from both the posterior mean and the posterior covariance (i.e., the uncertainty). Although the negative log-likelihood function is not directly used as the training loss, it

is exploited here as one additional validation metric to assess the DA skill of these models beyond using traditional path-wise measurements such as the MSE.

To demonstrate the performance of CGNSDE on various types of complex dynamical systems, three dynamical systems are utilized to generate the true signal for building and training the CGNSDE. Depending on the choice of the observed and unobserved state variables, the true system can be a CGNS or a non-CGNS. Here non-CGNS refers to any dynamical system that can not be written in the general form of (4.1). On the one hand, when the underlying system is a CGNS, then closed analytic formulae (4.2) can be exploited to compute the exact DA solution. Such a solution, as the optimal estimate for the state, will be compared with the ones from the CGNSDE models and the knowledge-based regression model. On the other hand, if the true system is not a CGNS, then the ensemble Kalman-Bucy filter (EnKBF) [72] will be utilized to provide an approximate optimal reference solution in assessing the DA performance of all the models.

A summary of the true underlying system and the conclusion of the associated numerical experiments is as follows.

1. The noisy Lorenz 84 system. It is a low-dimensional chaotic system that can be formalized as a CGNS. The study in Section 4.1.4.1 is utilized as a proof-of-concept of the skillful performance of the CGNSDE. By specifically prescribing imperfect knowledge-based components, the result shows that the additional neural network components significantly improve the model performance, and the DA loss is crucial in enhancing the DA skill.

2. The projected stochastic Burgers-Sivashinsky equation. It is a low-dimensional chaotic system that violates the assumption of CGNS and is featured by extreme events. The results in Section 4.1.4.2 show that the CGNSDE with the DA loss can accurately describe the model features and find the DA solution for such a non-CGNS. Notably, the CGNSDE with the DA loss reaches a posterior mean estimation comparable to the one by applying the EnKBF to the true system. The posterior covariance from the DA associated with the CGNSDE accurately characterizes the uncertainty. In addition, the extreme events are well captured by the CGNSDE in both the forecast and DA solutions.

3. The Lorenz 96 system. It is a moderate-dimensional chaotic system with local nonlinear interactions. An effective way of designing and training the low-dimensional

components of the neural network.

- Case 1: Homogeneous solution and CGNS. This case is designed to have a statistically homogeneous solution. By specifying the observed and unobserved variables, the system is formalized as a CGNS. The results show that the proposed CGNSDE framework has a good performance and scales well with the dimensionality of the system. In addition, the CGNSDE model trained with the DA loss can provide stable long-term simulations that have comparable statistics with the true system.
- Case 2: Homogeneous solution and non-CGNS: By choosing a different set of observed variables, the true system becomes a non-CGNS. The results confirm that the CGNSDE model trained with the DA loss can still handle the modeling and DA for a non-CGNS. The DA results are also comparable to the ones from true system with EnKBF.
- Case 3: Inhomogeneous solution and non-CGNS: The parameters are chosen to be spatial-dependent and therefore the solution demonstrates statistically inhomogeneous features. The results show that the CGNSDE model trained with the DA loss can still capture the true hidden states well. More specifically, the mean estimation from CGNSDE with DA has a good agreement with most system states, and the uncertainties are noticeably larger for those states with less satisfactory estimation of their means.

4.1.4.1 The Lorenz 84 Model: A Low-Order Chaotic System

The noisy Lorenz 84 model is a simple analogue of the global atmospheric circulation [102, 273], which has the following form [331, 332]:

$$\begin{aligned}
\frac{dx}{dt} &= -(y^2+z^2) - a(x-f) + \sigma_x \dot{W}_x, \\
\frac{dy}{dt} &= -bxz + xy - y + g + \sigma_y \dot{W}_y, \\
\frac{dz}{dt} &= bxy + xz - z + \sigma_z \dot{W}_z.
\end{aligned} \tag{4.16}$$

In (4.16), the zonal flow x represents the intensity of the mid-latitude westerly wind current. There is a wave component in the system, with y and z denoting the cosine and sine

phases, respectively, of a sequence of vortices superimposed on the zonal flow. The wave variables are scaled in relation to zonal flow such that $x^2+y^2+z^2$ is the total scaled energy encompassing kinetic, potential, and internal energy components. Note that these equations can be derived as a Galerkin truncation of the two-layer quasigeostrophic potential vorticity equations in a channel.

Viscous and thermal processes will linearly damp the vortices in this system. The parameter $a<1$ is a Prandtl number and the time unit of the system is defined by the damping time. The term af, proportional to the contrast between solar heating at low and high latitudes, is the external force that drives the zonal flow. The terms xy and xz illustrate how the wave is amplified through its interaction with the zonal flow. With the wave transporting heat poleward, the temperature gradient will be reduced at a rate proportional to the square of the amplitudes indicated by the term $-(y^2+z^2)$. If $x>0$, the terms $-bxz$ and bxy represent the wave westward displacement by the zonal current. With $b>1$, the displacement is allowed to overcome the amplification. A secondary forcing g mimics the contrasting thermal properties of the underlying surface of zonally alternating oceans and continents and therefore can affect the wave. For $g>0$ the system clearly shows chaotic behavior.

The parameter values used in the following tests are the standard values that create chaotic features:

$$a=\frac{1}{4},\quad b=4,\quad f=8,\quad g=1,\quad \sigma_x=1,\quad \sigma_y=0.05,\quad \text{and}\quad \sigma_z=0.05. \tag{4.17}$$

See Figure 4.1 for a model simulation, the associated chaotic behavior, and the equilibrium statistics. Note that the decorrelation time (i.e., the integration of the autocorrelation function (ACF)) of the x variable is about 0.5 time units, and those of the y and z variables are about 0.2 time units.

In the following test, y and z are assumed to be the observed variables while x is unobserved. Note that a larger noise in the process of x is taken to provide more variabilities and allow for the examination of the skill of state estimation. A time series of 250 units is generated, where the first 50 units are utilized for training, and the remaining 200 units are applied for testing.

As the Lorenz 84 is already a CGNS when y and z are treated as the observed variables, applying the causal inference approach in Section 4.1.3.1 will fully recover the exact dynam-

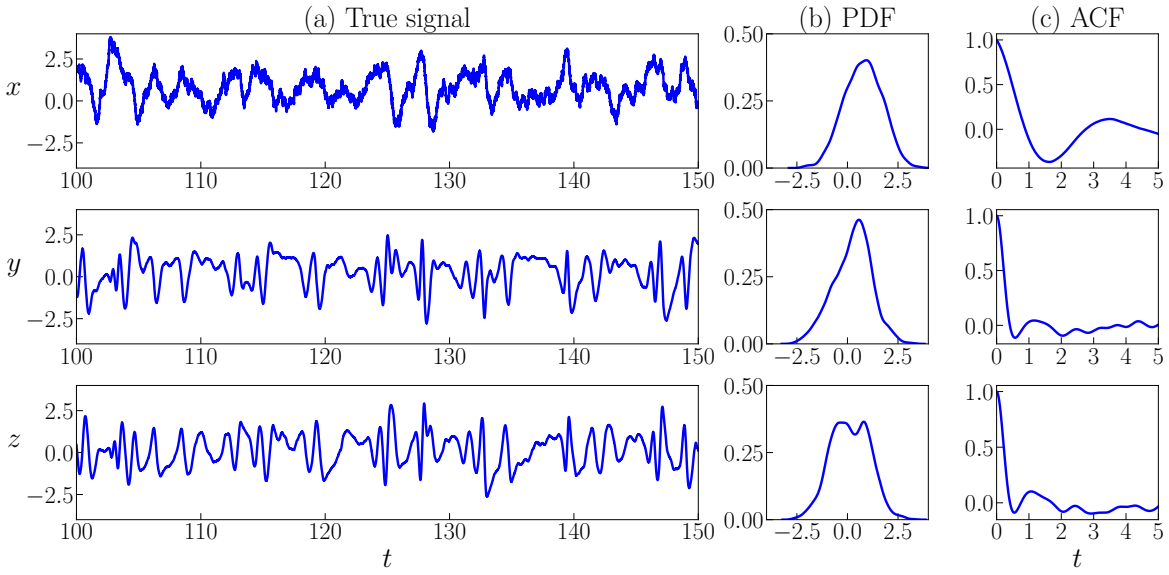

Figure 4.1: Model trajectories and the associated statistics of the true Lorenz 84 system. Panel (a): time series of each variable. Panel (b): the probability density function (PDF). Panel (c): the auto-correlation function (ACF). It should be noted that the PDFs and ACFs are estimated from much longer time series than the ones presented in Panel (a).

ics. The additional neural network components are no longer needed in such a case. Since this numerical test example aims to illustrate the effectiveness of the CGNSDE with the DA loss, the physically explainable components in the hybrid model are manually specified, which can be regarded as using the user's prior knowledge to determine the explainable components of the model structure. Some terms in the original system are removed on purpose, which allows some room for the neural networks to improve the results. Note that the exact DA solution of the Lorenz 84 is available using (4.2), which will serve as the reference solution to examine the performance of the CGNSDE.

With the prescribed knowledge-based components, the CGNSDE is given by:

$$\begin{aligned}
\frac{dx}{dt} &= f_x + a_x x + b_x z^2 + \mathbf{NN}_1(y,z;\theta) + \mathbf{NN}_4(y,z;\theta)x + \sigma_x \dot{W}_x, \\
\frac{dy}{dt} &= f_y + a_y y + \mathbf{NN}_2(y,z;\theta) + \mathbf{NN}_5(y,z;\theta)x + \sigma_y \dot{W}_y, \\
\frac{dz}{dt} &= f_z + a_z z + b_z xz + \mathbf{NN}_3(y,z;\theta) + \mathbf{NN}_6(y,z;\theta)x + \sigma_z \dot{W}_z,
\end{aligned} \tag{4.18}$$

where $\mathbf{NN}_i$ is the i-th output of a single neural network $\mathbf{NN}: \mathbb{R}^2 \mapsto \mathbb{R}^6$ which is parameterized

by θ. The feed-forward neural network utilized here has 5 layers with 508 parameters. The training settings are $N_s = 200$ (0.2 units), $N_l = 50000$ (50 units) and $N_b = 5000$ (5 units). The training starts with minimizing solely the forecast MSE (4.11) for the first 10000 epochs. Then the DA loss in (4.13) is included to minimize the total loss in (4.15) for the subsequent 500 epochs. The optimizer for all models is selected as Adam with a learning rate 10^{-3}.

Table 4.1 includes the performance of the models in a test period, which is different from the training data. The forecast MSE is based on 0.2 units prediction, while the DA MSE and DA negative log-likelihood are based on 200 units. It illustrates that the CGNSDE significantly outperforms the specified knowledge-based regression model regarding both state prediction and DA, indicating the critical role of the neural network components in the CGNSDE. The CGNSDE with the additional DA loss results in a smaller MSE in the posterior mean and a lower negative log-likelihood in the entire posterior distribution than the one with only the standard forecast MSE loss. This means the CGNSDE with the DA loss improves the DA skill with enhanced path-wise accuracy and reduced posterior uncertainty. Notably, the MSE and negative log-likelihood of the data assimilation solution using the true Lorenz 84 system are 0.0138 and -0.8175, respectively, similar to the result from the CGNSDE with the DA loss.

It is worth noticing that the CGNSDE without the DA loss is already quite skillful in DA. This is because the test model here is a CGNS. Therefore, even with only the forecast loss, the CGNSDE nearly captures the true dynamics, which leaves only a small room for the CGNSDE with the DA loss to improve the results further. In the following two numerical experiments, when the true system is not a CGNS, the advantage of the CGNSDE with the DA loss will become more significant, as the DA loss helps further improve the CGNSDE to characterize interdependence between different state variables.

Table 4.1: Lorenz 84: Performance of three models in the test period including knowledge-based regression model, CGNSDE without DA loss, and CGNSDE with the DA loss.

	Forecast MSE	DA MSE	DA Neg-Log-Likelihood
Knowledge-based regression model	0.2114	8.5685	20.8234
CGNSDE without DA loss	0.0487	0.0444	0.6289
CGNSDE with the DA loss	0.0469	0.0183	-0.5870

The findings in Table 4.1 are validated by Figure 4.2, which shows the DA results using the true system and three models as the forecast model, respectively. The two CGNSDEs

(with and without DA loss) significantly improve the performance of the DA compared with the pre-determined knowledge-based regression models, indicating the necessity of incorporating the neural network components in the CGNSDE. The CGNSDE with the DA loss can further improve the DA accuracy compared to its counterpart with only the standard forecast MSE as the loss function.

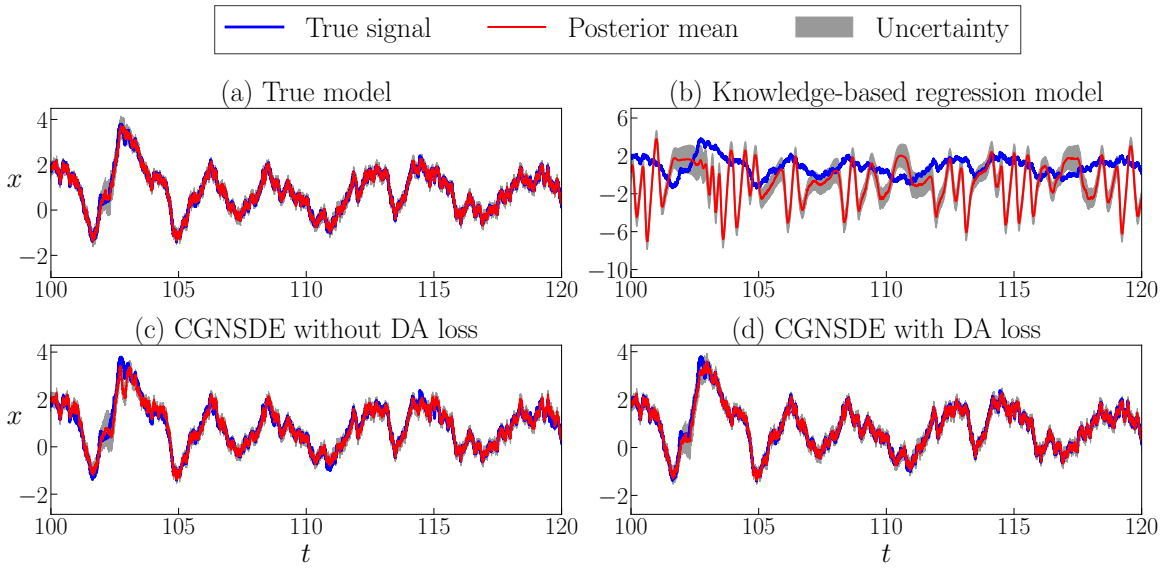

Figure 4.2: DA results of Lorenz 84 system from the closed analytic formulae in (4.2) for true system and three models. The uncertainties are indicated by the grey colored regions, which correspond to two standard deviation from the posterior mean.

Finally, Figure 4.3 shows one long-term simulation of the CGNSDE with the DA loss. Panel (a) indicates that the CGNSDE can provide stable long-term simulations without explicitly regularizing its stability in training. The overall patterns of the simulated time series are similar to the true system results. Panels (b) and (c) show that the long-term statistics (e.g., the PDFs and the ACFs) associated with the simulations from the CGNSDE have a qualitative agreement with the true system, despite utilizing only short time series as the forecast loss in the training stage.

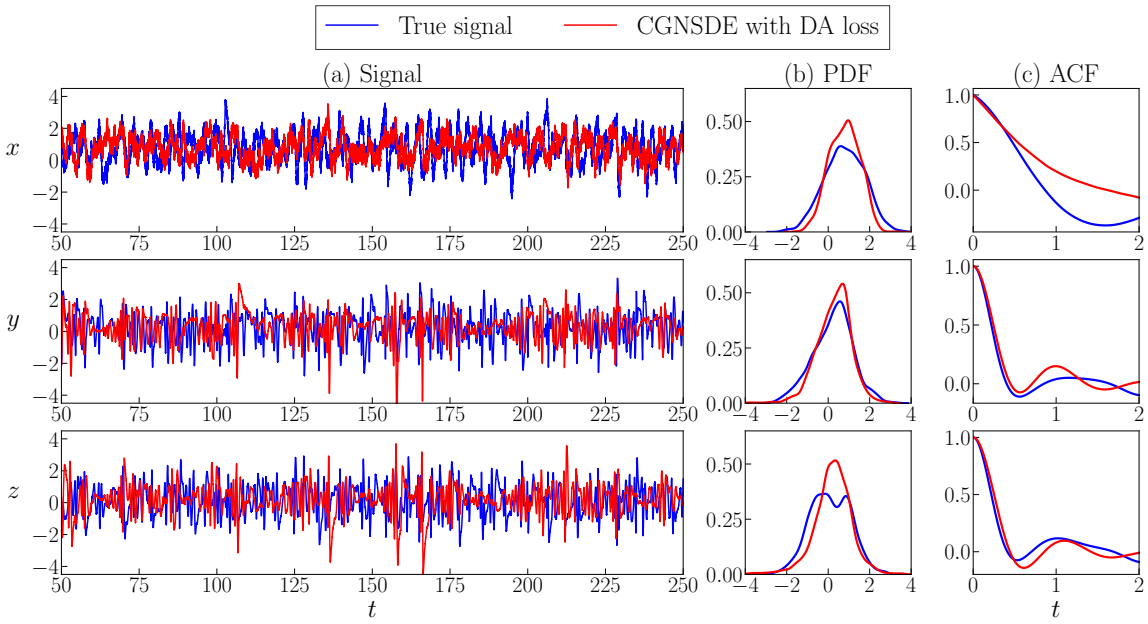

Figure 4.3: One long-term realization of the CGNSDE model trained with both forecast and DA losses for the Lorenz 84 system. Panel (a): time series of different variables. Panel (b) and (c): the associated PDFs and ACFs.

4.1.4.2 The Projected Stochastic Burgers–Sivashinsky Equation: A Highly Nonlinear System with Intermittency

The Fourier-Galerkin projection of stochastic Burgers-Sivashinsky equation is a three-dimensional SDEs with energy-conserving quadratic nonlinear terms and subject to additive white noise forcing [333, 334]:

$$\begin{aligned}
\frac{dx}{dt} &= \beta_x x + \alpha xy + \alpha yz + \sigma_x \dot{W}_x, \\
\frac{dy}{dt} &= \beta_y y - \alpha x^2 + 2\alpha xz + \sigma_y \dot{W}_y, \\
\frac{dz}{dt} &= \beta_z z - 3\alpha xy + \sigma_z \dot{W}_z,
\end{aligned} \tag{4.19}$$

where the coefficients for the linear terms are chosen such that β_x is positive to introduce linear instability into the system, while β_y and β_z are negative, representing linear damping effects. The coefficient $\alpha > 0$ controls the strength of the nonlinearity. The noise strength coefficients σ_x, σ_y, and σ_z are positive constants. This system can, for instance, be obtained

as a Fourier-Galerkin projection of the stochastic Burgers-Sivashinsky equation

$$\frac{\partial u}{\partial t} = \left(\nu \partial_{xx} u + \lambda u - u \partial_x u\right) + \dot{W}(t, x)$$

posed on a bounded interval $x \in (0, L)$ subject to homogeneous Dirichlet boundary conditions. In this context, β_x, β_y, and β_z are simply the three largest eigenvalues of the linear operator, and α is linked to the domain size L via $\alpha = \pi/(\sqrt{2}L^{3/2})$.

In the following, the largest-scale variable x is treated as the observed variable while there are no direct observations for y and z. Under this splitting of the state variables, system (4.19) does not have the conditional Gaussian structure due to the quadratic nonlinear term αyz between the unobserved variables that appear in (4.19). Nevertheless, due to the low dimensionality, the ensemble Kalman-Bucy filter (EnKBF) [72] can be utilized to provide a reference solution of DA.

The parameter values used in the following tests are the standard values that create strongly intermittent features with extreme events and highly non-Gaussian PDFs:

$$\beta_x = 0.2, \quad \beta_y = -0.3, \quad \beta_z = -0.5, \quad \alpha = 5, \quad \sigma_x = 0.3, \quad \sigma_y = 1, \quad \text{and} \quad \sigma_z = 1. \tag{4.20}$$

A time series of 600 units is generated, where the first 100 units are utilized for training, and the remaining 500 units are applied for testing. Figure 4.4 shows a period of the model simulation and the associated statistics. The time series display strong intermittent behavior with non-Gaussian statistics and extreme events.

To develop a CGNSDE, the causal inference method described in Section 4.1.3.1 is utilized to identify the knowledge-based components of system dynamics. The library is constructed with candidate functions containing all possible linear and quadratic nonlinear functions that allow the model to satisfy the conditional Gaussian structures. Therefore, the library is given by $\{x, y, z, x^2, xy, xz\}$. The values of the resulting causation entropies from these candidate functions to the target dynamics are included in Table 4.2. It is worth noting that the relative strengths of the causation entropies should be compared only within each row. Different rows, presenting the governing equations of different state variables, are independent of the others. The causation entropies in different rows may have significant differences in the order of amplitudes. By considering the relative magnitudes within each row, the candidate functions (z, xy) are selected for the x dynamics, (x^2, xz) are for the y

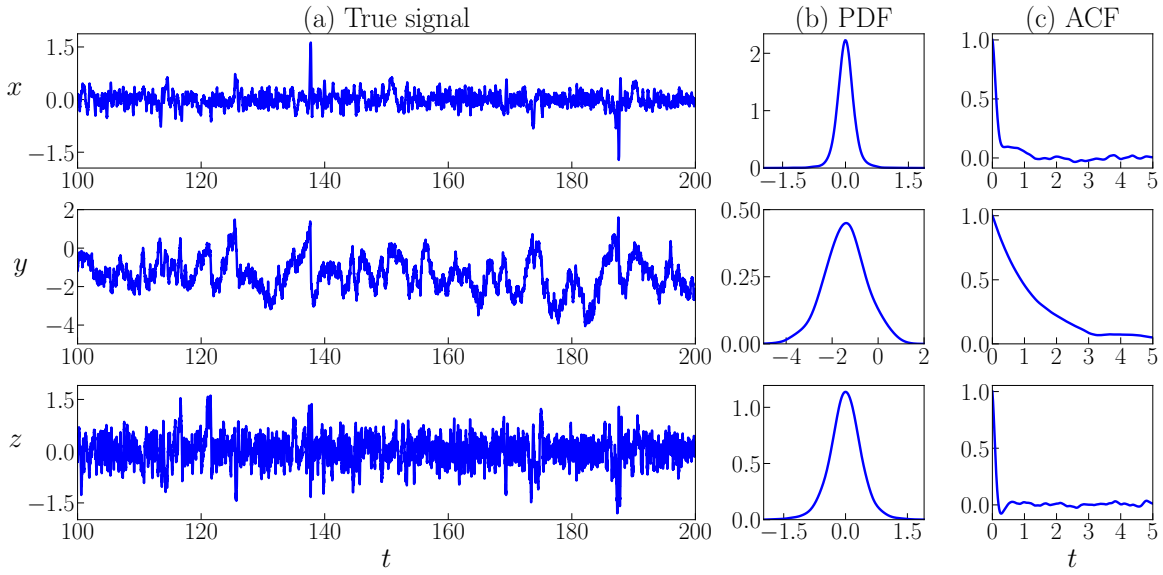

Figure 4.4: One simulation of the projected stochastic Burgers–Sivashinsky equation (4.19). Panel (a): time series of each state variable. Panel (b): the PDFs. Panel (c): the ACFs. It should be noted that the PDFs and ACFs are estimated from much longer simulations than the one presented in Panel (a).

dynamics, and xy is for the z dynamics.

Table 4.2: The projected stochastic Burgers–Sivashinsky equation: causation entropy between dynamics and candidate functions in the library. The significant values, corresponding to the terms used to build the knowledge-based components in the CGNSDE, are highlighted in bold font. Note that the relative strengths of the causation entropies should be compared only within each row.

	x	y	z	x^2	xy	xz
$\dot{x}$	0.000	0.000	**0.093**	0.000	**0.049**	0.001
$\dot{y}$	0.000	0.000	0.000	**0.004**	0.000	**0.011**
$\dot{z}$	0.000	0.000	0.001	0.000	**0.070**	0.000

After supplementing neural networks to the knowledge-based regression model, the

CGNSDE for this projected stochastic Burgers–Sivashinsky equation reads:

$$\begin{aligned}
\frac{dx}{dt} &= a_x z + c_x xy + \mathbf{NN}_1(x;\theta) + \mathbf{NN}_4(x;\theta)y + \mathbf{NN}_5(x;\theta)z + \sigma_x \dot{W}_x \\
\frac{dy}{dt} &= b_y x^2 + c_y xz + \mathbf{NN}_2(x;\theta) + \mathbf{NN}_6(x;\theta)y + \mathbf{NN}_7(x;\theta)z + \sigma_y \dot{W}_y \\
\frac{dz}{dt} &= c_z xy + \mathbf{NN}_3(x;\theta) + \mathbf{NN}_8(x;\theta)y + \mathbf{NN}_9(x;\theta)z + \sigma_z \dot{W}_z
\end{aligned} \tag{4.21}$$

In the CGNSDE (4.21), $\mathbf{NN}_i$ is the i-th output of a single neural network $\mathbf{NN} : \mathbb{R}^1 \mapsto \mathbb{R}^9$ which is parameterized by θ. The feed-forward neural network utilized here has 5 layers with 449 parameters.

The setup for training the two CGNSDEs is: $N_s = 50$ (0.5 units), $N_l = 10000$ (100 units), $N_b = 1000$ (10 units). The CGNSDE without DA is trained for 10000 epochs to minimize the forecast MSE (4.11). The CGNSDE with DA is achieved by retraining CGNSDE for another 500 epochs to minimize total loss (4.15) including DA loss. The optimizer for all models is selected as Adam with a learning rate of 10^{-3}.

The test results of all three models are summarized in Table 4.3. In the test period, the forecast MSE is calculated by state prediction with 0.5 units, while the DA MSE and DA negative log-likelihood are based on 500 units. Although the CGNSDE trained without the DA loss gives the lowest error for short-term prediction, which is around the level of the intrinsic error due to the random noise in the true system, it has the largest DA MSE. The error is even more significant than the knowledge-based regression model. In contrast, the CGNSDE model trained with both the forecast and the DA losses in (4.15) can significantly enhance the DA performance with a slight trade-off of short-term forecast accuracy. This improvement indicates the essential role of including the DA loss in training the CGNSDE.

It should be noted that the true system is a non-CGNS in this example, and thus, the DA using the true system cannot be done by applying the explicit formulae in (4.2). Therefore, the EnKBF is adopted to obtain the DA solution of the true system as a reference. The number of ensemble members in the EnKBF is set to be $J = 100$. Using the original noise parameters ($\sigma_x = 0.3, \sigma_y = 1, \sigma_z = 1$), the EnKBF will encounter a catastrophic filter divergence [335], i.e., the solution has a finite-time numerical blow-up issue. By applying a standard noise inflation strategy [336], the minimum DA MSE and the corresponding negative log-likelihood of applying the EnKBF to the true system are 0.2237 and 7.3375, respectively. Notably, the DA MSE of the CGNSDE with the DA loss is 0.2674, which is comparable with the solution

given by EnKBF. Since the CGNSDE framework does not demand ensemble simulations, it can reduce the computational cost and avoid potential sampling errors due to an insufficient ensemble size.

Table 4.3: The projected stochastic Burgers–Sivashinsky equation: Performance in the test period of the knowledge-based regression model, the CGNSDE without DA loss, and the CGNSDE with the DA loss.

	Forecast MSE	DA MSE	DA Neg-Log-Likelihood
Knowledge-based regression model	0.1799	0.8589	2.9459
CGNSDE without DA loss	0.1347	1.2907	4.4199
CGNSDE with the DA loss	0.1446	0.2674	1.3528

Figure 4.5 compares the DA results by applying the EnKBF to the true system and by applying the closed analytic formulae to the regression model and the two CGNSDEs. Panel (a) indicates that the results of EnKBF for the true system still have a noticeable difference from the true states y and z. Despite tuning the noise coefficients to minimize the error in the posterior mean, the posterior uncertainty seems severely underestimated. In contrast, the posterior mean estimates from the CGNSDE trained with the DA loss, as shown in Panel (d), demonstrate a similar performance as the EnKBF for the true system, while the range of the associated uncertainty of the CGNSDE with the DA loss can cover the truth, including the extreme events. The results confirm the advantages of the CGNSDE framework with DA for systems with highly non-Gaussian features. On the other hand, both the regression model and the CGNSDE without the DA loss lead to much larger biases in the DA solutions.

Figure 4.6 compares the long-term simulation results of the CGNSDE with the DA loss to the true signal. Similar to the previous example, the CGNSDE can also provide effective long-term simulations, demonstrating a qualitative agreement with the time series of the true system as presented in Panel (a). Remarkably, the long-term time series of the modeled data can reproduce the overall pattern as the true signal with noticeable extreme events in the x and z processes. Panels (b) and (c) show the associated PDFs and the ACFs. The PDFs of x and y from the CGNSDE are nearly identical to the truth, especially in capturing the non-Gaussian features. The PDF of z has a slightly heavier tail. Note that the model is trained based only on short data. The skillful long-term forecast result is partially due to the help from the additional DA loss, which highlights the dynamical coupling between different state variables that advances the long-term behavior of the system. Similarly, the ACFs from the CGNSDE are nearly the same as the truth. In contrast, the long-term

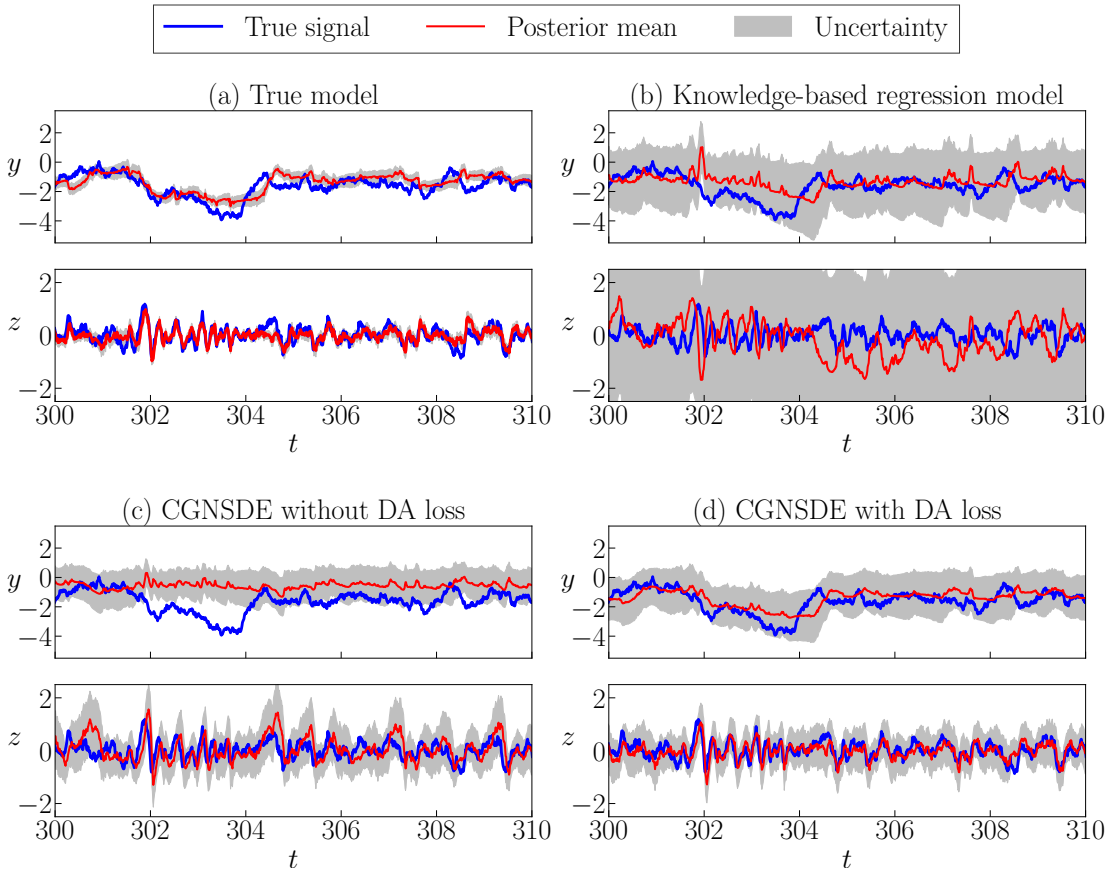

Figure 4.5: DA results of the projected stochastic Burgers–Sivashinsky equation from EnKBF for true system and closed analytic formulae in (4.2) for three models. The uncertainties are indicated by the grey colored regions, which correspond to two standard deviations from the posterior mean.

simulation from the knowledge-based model has highly oscillated patterns in the y and z dynamics, while the one from CGNSDE without DA loss cannot reproduce the extreme events in x dynamics (not shown here).

4.1.4.3 The Lorenz 96 Model: A Moderate-Dimensional Chaotic System

The Lorenz 96 model with noise is given by [253, 337]

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - c_i x_i + F_i + \sigma_i \dot{W}_i, \quad i = 1, 2, 3, \ldots, I. \tag{4.22}$$

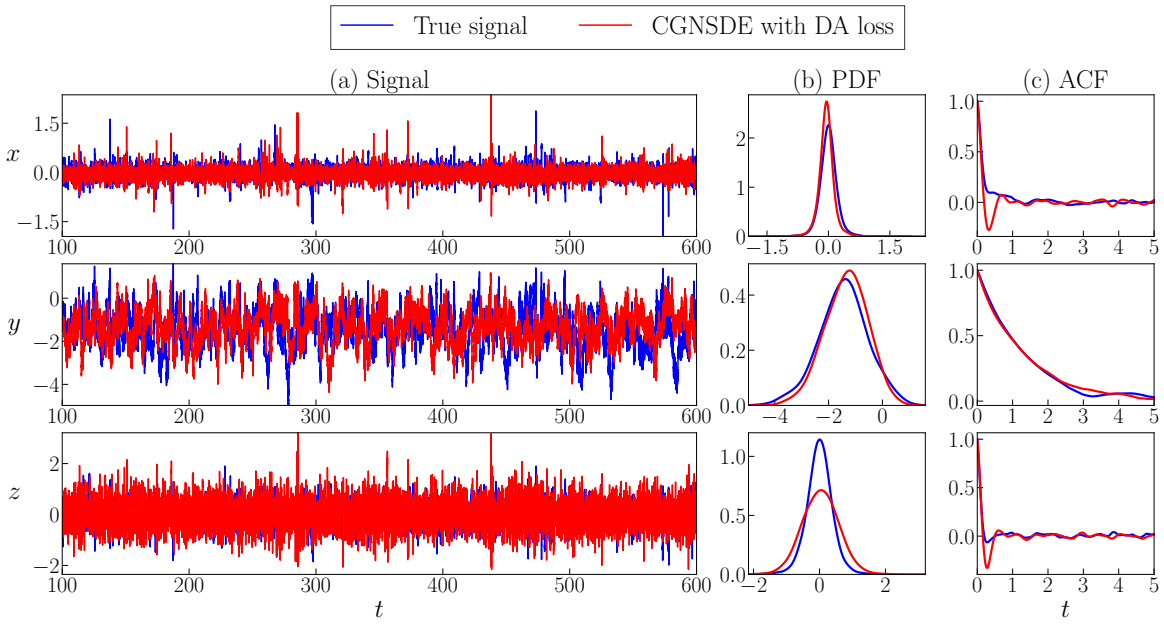

Figure 4.6: Long-term simulations of the CGNSDE with the DA loss starting from the initial state of test data for the projected stochastic Burgers–Sivashinsky equation. Panel (a): time series of each state. Panels (b) and (c): PDFs and ACFs.

The model can be regarded as a coarse discretization of atmospheric flow on a latitude circle with complicated wave-like and chaotic behavior. It schematically describes the interaction between small-scale fluctuations with larger-scale motions. It is widely used as a testbed for DA, state forecast, uncertainty quantification, and parameterization in numerical weather forecasting [254, 256]. Notably, depending on the choice of the set of the observed and unobserved variables, the Lorenz 96 equation (4.22) can be a CGNS or a non-CGNS. The study based on the Lorenz 96 model includes three cases:

1. The true dynamics is a CGNS with spatially homogeneous statistics. The system has constant parameters such that the spatial patterns are statistically homogeneous. In other words, different grids, indicated by different indices $\mathfrak{i}$ in (4.22), have the same equilibrium statistics. Two-thirds of the states are observable, and the other one-third are unobserved. In such a way, the true model becomes a CGNS. The primary purpose of this test is to demonstrate the capability of the CGNSDE model in handling relatively high-dimensional systems.

2. The true dynamics is a non-CGNS with spatially homogeneous statistics. The same

spatially homogeneous system is utilized as in Case 1. Half of the states are observable, and the other half are unobserved, so the true system becomes a non-CGNS. The focus of this case is on demonstrating the capability of the CGNSDE in handling a non-CGNS with a relatively high dimension.

3. The true dynamics is a non-CGNS with spatially inhomogeneous statistics. The system has spatially dependent parameters, so the statistical behavior at different grid points is inhomogeneous. The true system is a non-CGNS, with half of the states being observable and the other half being unobservable. The main focus of this case is on demonstrating that the performance of using the CGNSDE with a simple translate-invariant neural network structure can handle spatially inhomogeneous dynamics.

The following parameter values are adopted as in the true system for the first two cases with homogeneous dynamics:

$$I = 36, \qquad F_i = 8, \qquad c_i = 1, \qquad \text{and} \qquad \sigma_i = 0.5, \tag{4.23}$$

and the parameters for the third case with inhomogeneous statistics are as follows,

$$I = 36, \qquad F_i = 8, \qquad c_i = 2 + 1.5\sin(2\pi(i-1)/I), \qquad \text{and} \qquad \sigma_i = 0.5. \tag{4.24}$$

A time series of 300 units is generated, where the first 100 units are utilized for training, and the remaining 200 units are applied for testing.

It is worth highlighting that the nonlinearity in Lorenz 96 is given by advection. Due to its localized interactions between state variables, the complexity of the neural network part can be significantly reduced. In particular, the number of neural networks will not increase as a function of the dimension of the underlying system.

Case 1: CGNS that is spatially homogeneous

Figure 4.7 includes a model simulation and the associated statistics of the true Lorenz 96 system (4.22) with parameters in (4.23). Panel (a) shows the spatiotemporal patterns (i.e., the Hovmoller diagram), where the propagation of waves and the chaotic pattern of the system can be seen. Panel (b) shows the time series of x_1, where the associated PDFs and

ACFs are displayed in Panels (c)–(d). The behavior at other grid points is similar since the system is statistically homogeneous in space.

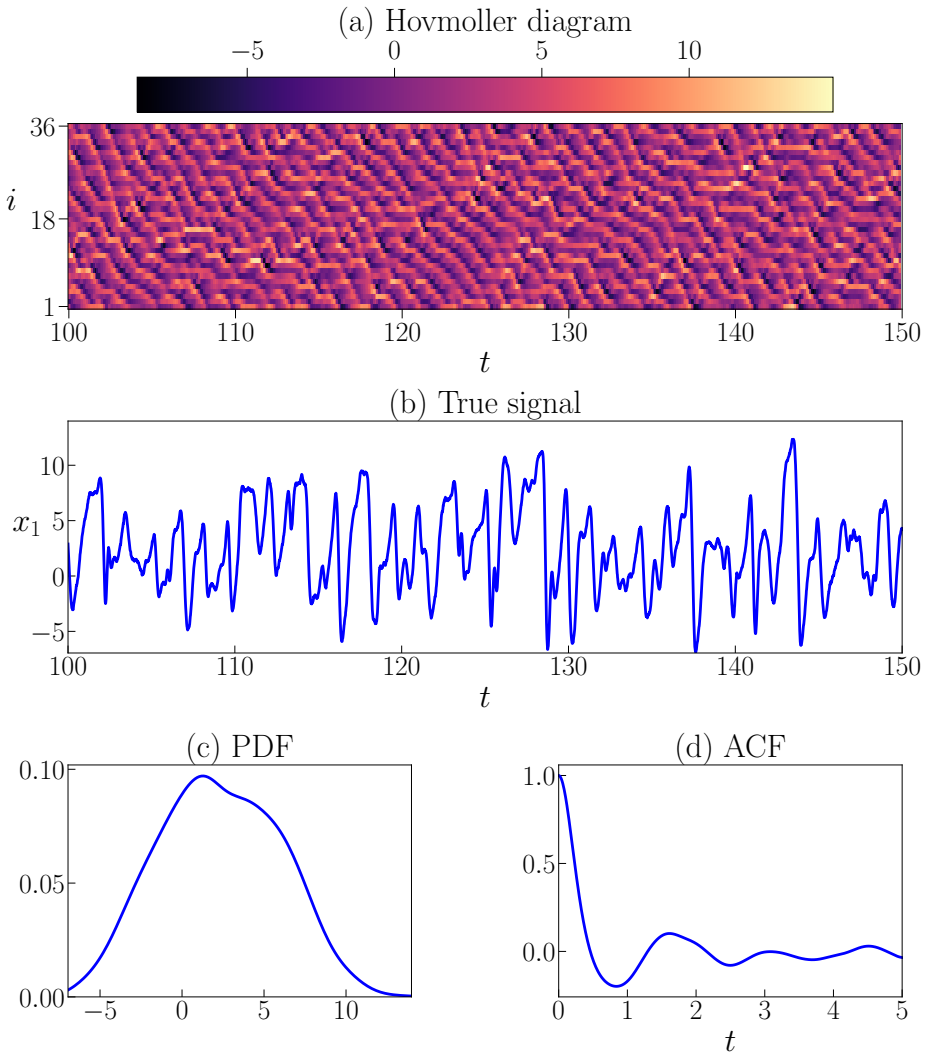

Figure 4.7: Model simulation and statistics of the noisy Lorenz 96 system with parameters in (4.23). Panel (a): the Hovmoller diagram of the spatiotemporal patterns. Panel (b): time series of x_1. Panel (c): the PDF of x_1. Panel (d): the ACF of x_1. Note that the PDF and ACF are estimated from a simulation longer than the one presented in Panel (b). The behavior at other grid points is similar since the system is statistically homogeneous in space.

In this case, the observed state variables are $\mathbf{u}_1 = [x_1, x_2, x_4, x_5, \ldots, x_{34}, x_{35}] \in \mathbb{R}^{24}$ and the unobserved variables are $\mathbf{u}_2 = [x_3, x_6, \ldots, x_{36}] \in \mathbb{R}^{12}$. With such a choice, the Lorenz 96 system can be formalized as a CGNS in (4.1).

Since the Lorenz 96 model in such a case is already a CGNS, if the causal inference is applied, then the knowledge-based model will fully recover the true dynamics. As in the Lorenz 84 test case, an intrinsic bias in the prior knowledge is introduced on purpose to allow neural networks to play some roles in improving the model behavior. To this end, the candidate function library is assumed only to contain linear functions. Therefore, applying causal inference leads to the part of the knowledge-based regression model with the correct linear structure of the original system. The remaining information of the system will be characterized by neural networks.

With the linear terms identified by the causal inference, the CGNSDE reads

$$\frac{dx_i}{dt} = \begin{cases} \text{if } i \bmod 3 = 1: \\ \quad \tilde{F} + \tilde{c}x_i + \mathbf{NN}_1^{(1)}(x_{i-2}, x_i, x_{i+1}; \theta_1) + \\ \qquad\qquad \mathbf{NN}_2^{(1)}(x_{i-2}, x_i, x_{i+1}; \theta_1)x_{i-1} + \\ \qquad\qquad \mathbf{NN}_3^{(1)}(x_{i-2}, x_i, x_{i+1}; \theta_1)x_{i+2} + \sigma_i \dot{W}_i, \\ \text{if } i \bmod 3 = 2: \\ \quad \tilde{F} + \tilde{c}x_i + \mathbf{NN}_1^{(2)}(x_{i-1}, x_i, x_{i+2}; \theta_2) + \\ \qquad\qquad \mathbf{NN}_2^{(2)}(x_{i-1}, x_i, x_{i+2}; \theta_2)x_{i-2} + \\ \qquad\qquad \mathbf{NN}_3^{(2)}(x_{i-1}, x_i, x_{i+2}; \theta_2)x_{i+1} + \sigma_i \dot{W}_i, \\ \text{if } i \bmod 3 = 0: \\ \quad \tilde{F} + \tilde{c}x_i + \mathbf{NN}_1^{(3)}(x_{i-2}, x_{i-1}, x_{i+1}, x_{i+2}; \theta_3) + \\ \qquad\qquad \mathbf{NN}_2^{(3)}(x_{i-2}, x_{i-1}, x_{i+1}, x_{i+2}; \theta_3)x_i + \sigma_i \dot{W}_i, \end{cases} \tag{4.25}$$

where "mod" denotes the standard modulo operation that calculates the remainder, and $i = 1, 2, \ldots, 36$. The $\mathbf{NN}_j^{(i)}$ in (4.25) stands for the j-th output of i-th neural network. The CGNSDE is developed by taking advantage of the fact that state variables only have localized dependence. For $i \bmod 3 = 1$, x_i are the observed variables, and their nearby dependent state variables are $(x_{i-2}, x_{i-1}, x_{i+1}, x_{i+2})$, in which x_{i-1} and x_{i+2} are unobserved. Therefore, the first part of (4.25) is constructed to satisfy the CGNS form, with linear dependency on nearby hidden states x_{i-1} and x_{i+2}. The second part of (4.25) can be constructed similarly, while the third part has a different form from the previous two, mainly because x_i are unobserved state variables for $i \bmod 3 = 0$. The three neural networks all have 3 layers with 57, 57, and 64 parameters, respectively.

It is worth noting that the same neural network (with the same structures and parameters) is utilized for each set of state variables with different indices i. For example, the first neural network $\mathbf{NN}^{(1)}$ appears in the governing equations of $x_1, x_4, \ldots, x_{34}$. This is due to the translate-invariant structure of the Lorenz 96 system, which is assumed to be a known feature in developing the CGNSDE. By applying such a physical property, the complexity of the neural network part can be significantly reduced since, otherwise, the number of neural networks will scale as a function of the dimension of the underlying system. Furthermore, when the statistics are homogeneous, the effective length of the training data equals the actual length of the observations multiplied by the number of grids. This means that with a large number of grids, a short segment of the model observations is usually sufficient to train the neural network.

The CGNSDE without the DA loss has been trained by minimizing the forecast MSE in (4.11) with $N_s = 1$. This training method based on one-step state prediction is formalized by training the model to approximate the mapping from states to dynamics in practice. Then the CGNSDE with the DA loss will be retrained by $N_s = 5$, $N_l = 10000$ and $N_b = 500$, which means the matching time of states prediction is 0.05 time units and the DA generation time is 100 time units with the first 5 time units being omitted as the burn-in period for DA. The Adam optimizer is selected with 10^{-3} learning rate.

The performance of the three models in the test period is summarized in Table 4.4. The forecast MSE is calculated by 0.2 units of state prediction, while the DA MSE and DA negative log-likelihood are based on 200 units. The knowledge-based model with only simple linear structures fails to approximate the underlying dynamics from data, resulting in substantial errors in both forecast and DA. Compared to this knowledge-based regression model, the CGNSDE without DA loss can significantly improve the forecast and DA performance after introducing the neural network components. By further incorporating DA loss in training, the CGNSDE slightly reduces the DA error in the test period but somewhat increases the forecast error. Notably, the MSE and negative log-likelihood of the data assimilation solution using the true Lorenz 96 system are 0.05284 and 1.0723, respectively, similar to the results from the two CGNSDEs. The comparable performance of the two CGNSDEs is not too surprising, since the true system is a CGNS so the CGNSDE with only the forecast loss may have already fully captured the true underlying dynamics. However, as seen below, this will not be the case when the true system is a non-CGNS.

Figure 4.8 shows the long-term simulation of CGNSDE with the DA loss and compares the

Table 4.4: Lorenz 96 system (Case 1): Performance of the models in the test period.

	Forecast MSE	DA MSE	DA Neg-Log-Likelihood
Knowledge-based regression model	3.8283	13.7121	33.8079
CGNSDE without DA loss	0.0410	0.0764	3.9665
CGNSDE with the DA loss	0.0506	0.0690	2.9490

results with the true system. As shown in Panel (a), the CGNSDE model can produce stable long-term simulation, which captures the wave propagation and the chaotic pattern of the underlying dynamics. Since the CGNSDE model in (4.25) has different types of expressions, one state of each kind is presented in Panel (b). The model successfully captures the overall patterns of the true time series, with a slight overestimate of the amplitude in x_1 and x_2. Panels (c) and (d) show the associated PDFs and the ACFs, confirming that the long-term statistics from the CGNSDE model are overall comparable to the truth, even though the forecast loss in the training period is based only on a much shorter time series data.

Case 2: Non-CGNS that is spatially homogeneous

In this case, $\mathbf{u}_1 = [x_1, x_3, x_5, \ldots, x_{35}] \in \mathbb{R}^{18}$ are the observed variables and $\mathbf{u}_2 = [x_2, x_4, \ldots, x_{36}] \in \mathbb{R}^{18}$ are unobserved. By doing so, the true Lorenz 96 system $[\mathbf{u}_1, \mathbf{u}_2]^\mathsf{T}$ becomes a non-CGNS.

The causal inference method is utilized to identify significant knowledge-based components. The function libraries (4.6) are different for state variables $\mathbf{u}_1$ and $\mathbf{u}_2$ to guarantee the overall conditional Gaussian structures. The results of causation entropy are summarized in Table 4.5. The selected quadratic basis for $x_i \in \mathbf{u}_1$ are $[x_i, x_{i+1}, x_{i-2}x_{i-1}, x_{i-1}x_{i+2}, x_i x_{i+1}]$, while for $x_i \in \mathbf{u}_2$ are $[x_i, x_{i-2}x_{i-1}, x_{i-1}x_{i+1}]$.

Table 4.5: Lorenz 96 system (Case 2): causation entropy from the candidate functions contributing to the dynamics. The significant values are highlighted in bold font.

$\dot{x}_i$, $i \bmod 2 = 1$	x_{i-2}	x_{i-1}	x_i	x_{i+1}	x_{i+2}	x_{i-2}^2	x_i^2	x_{i+2}^2	$x_{i-2}x_{i-1}$
	0.000	0.007	**0.013**	**0.025**	0.001	0.006	0.003	0.001	**0.247**
	$x_{i-2}x_i$	$x_{i-2}x_{i+1}$	$x_{i-2}x_{i+2}$	$x_{i-1}x_i$	$x_{i-1}x_{i+2}$	$x_i x_{i+1}$	$x_i x_{i+2}$	$x_{i+1}x_{i+2}$	
	0.003	0.007	0.003	0.008	**0.014**	**0.015**	0.004	0.001	

$\dot{x}_i$, $i \bmod 2 = 0$	x_{i-2}	x_{i-1}	x_i	x_{i+1}	x_{i+2}	x_{i-1}^2	x_{i+1}^2	$x_{i-2}x_{i-1}$	$x_{i-2}x_{i+1}$
	0.000	0.000	**0.085**	0.000	0.000	0.001	0.000	**0.820**	0.000
	$x_{i-1}x_i$	$x_{i-1}x_{i+1}$	$x_{i-1}x_{i+2}$	$x_i x_{i+1}$	$x_{i+1}x_{i+2}$				
	0.000	**0.726**	0.000	0.000	0.000				

With the selected candidate functions, the knowledge-based regression model (KRM) is

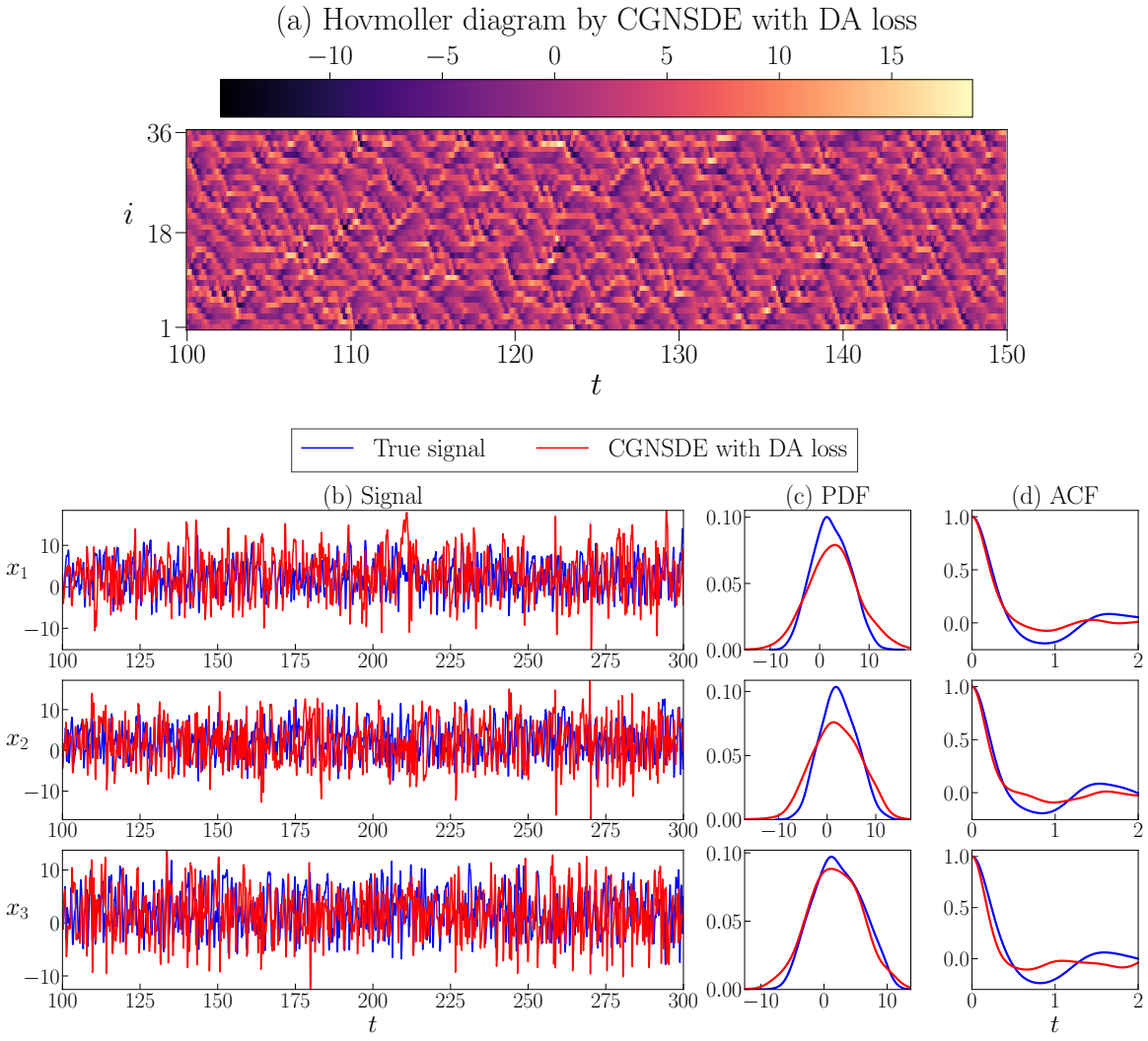

Figure 4.8: Long-term simulation results of CGNSDE with the DA loss for the Lorenz 96 system (Case 1). Panel (a): Hovmoller diagram of all states. Panel (b): time series of the first three states. Panel (c) and (d): probability density function (PDF) and auto-correlation function (ACF).

given by:

$$\frac{dx_i}{dt} = \begin{cases} \text{if } i \bmod 2 = 1: \\ \quad \mathbf{KRM}_1 = a_0 + a_1 x_i + a_2 x_{i+1} + a_3 x_{i-2} x_{i-1} + a_4 x_{i-1} x_{i+2} + a_5 x_i x_{i+1} + \sigma_i \dot{W}_i \\ \text{if } i \bmod 2 = 0: \\ \quad \mathbf{KRM}_2 = b_0 + b_1 x_i + b_2 x_{i-2} x_{i-1} + b_3 x_{i-1} x_{i+1} + \sigma_i \dot{W}_i. \end{cases} \tag{4.26}$$

After introducing the neural network components, the CGNSDE can be written in the form:

$$\frac{dx_i}{dt} = \begin{cases} \text{if } i \bmod 2 = 1: \\ \quad \mathbf{KRM}_1 + \mathbf{NN}_1^{(1)}(x_{i-2}, x_i, x_{i+2}; \theta_1) + \\ \qquad\qquad \mathbf{NN}_2^{(1)}(x_{i-2}, x_i, x_{i+2}; \theta_1) x_{i-1} + \\ \qquad\qquad \mathbf{NN}_3^{(1)}(x_{i-2}, x_i, x_{i+2}; \theta_1) x_{i+1}, \\ \text{if } i \bmod 2 = 0: \\ \quad \mathbf{KRM}_2 + \mathbf{NN}_1^{(2)}(x_{i-1}, x_{i+1}; \theta_2) + \\ \qquad\qquad \mathbf{NN}_2^{(2)}(x_{i-1}, x_{i+1}; \theta_2) x_{i-2} + \\ \qquad\qquad \mathbf{NN}_3^{(2)}(x_{i-1}, x_{i+1}; \theta_2) x_i + \\ \qquad\qquad \mathbf{NN}_4^{(2)}(x_{i-1}, x_{i+1}; \theta_2) x_{i+2}, \end{cases} \tag{4.27}$$

where again the localization is imposed, i.e., dx_i/dt only depends on nearby states $(x_i, x_{\pm 1}, x_{\pm 2})$. The $\mathbf{NN}_i^{(i)}$ in (4.27) stands for the j-th output of i-th neural network. The two neural networks have 5 layers with 438 parameters and 3 layers with 70 parameters, respectively.

With the same training and testing settings listed in Case 1, the test results of three models are summarized in Table 4.6. The CGNSDE without DA loss can outperform the knowledge-based regression model in both state forecast and DA. After retraining with the DA loss, the CGNSDE can further reduce the DA MSE and the negative log-likelihood at the expense of a slight increment in the forecast MSE. The DA MSE and DA negative log-likelihood of the reference solution from the EnKBF with the true system are 0.0771 and 13.9382, respectively.

Figure 4.9 compares the DA results of the unobserved state x_2 from the regression

Table 4.6: Lorenz 96 system (Case 2): Performance of the models in the test period.

	Forecast MSE	DA MSE	DA Neg-Log-Likelihood
Knowledge-based regression model	1.1137	1.3592	58.9987
CGNSDE without DA loss	0.7952	0.9710	54.5143
CGNSDE with the DA loss	0.9482	0.5763	30.0057

model and the two CGNSDEs to the true system using the EnKBF. Although the CGNSDE with the DA loss is still slightly worse than the perfect system, it is more skillful than the regression model and the CGNSDE without the DA loss. It is also worth noting that, despite some errors, the regression model gives an acceptable result. This demonstrates that the explainable knowledge-based components determined by the causation entropy indeed play an essential role in modeling the system and DA.

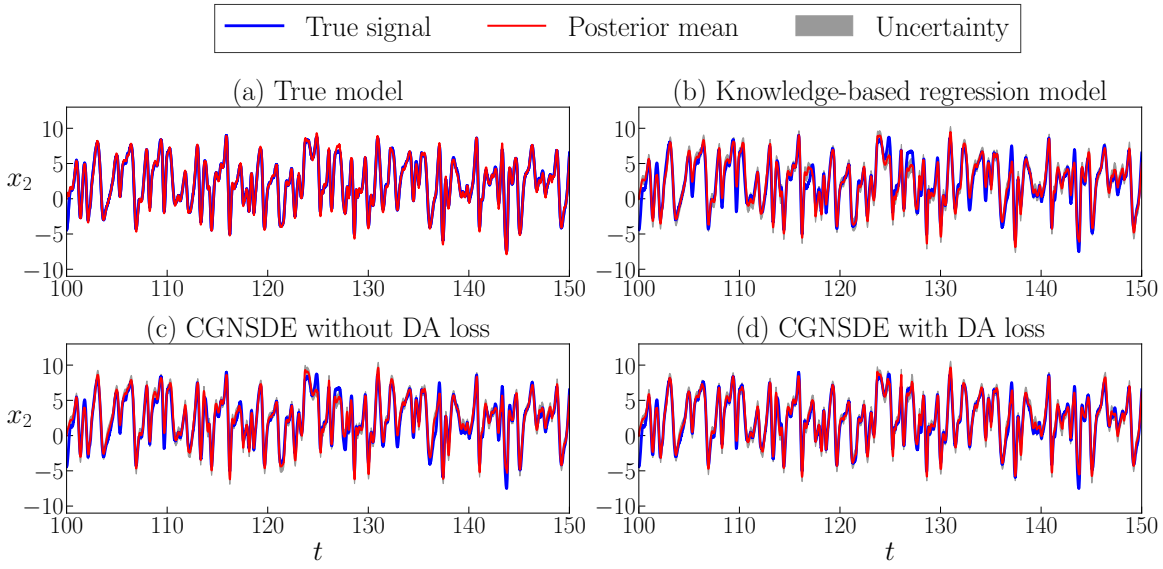

Figure 4.9: DA results of the unobserved state x_2 in the Lorenz 96 system (Case 2) for the true system (using the EnKBF) and for the other three models. The uncertainties are indicated by the grey colored regions, which correspond to two standard deviation from the posterior mean.

Figure 4.10 shows the long-term simulation results of CGNSDE with the DA loss. Since the CGNSDE has very different dynamics from the truth of such a highly chaotic system and is trained based only on short-term forecast loss, it is not expected that the CGNSDE will fully capture the dynamical features of the truth. Nevertheless, Panel (a) illustrates

that the simulation from the CGNSDE preserves the chaotic features and irregular wave propagations. The time series, as shown in Panel (b), validates the chaotic nature of the simulation from the CGNSDE. Although the statistics shown in Panels (c) and (d) contain some error, the overall variability and temporal dependence are reproduced by the CGNSDE.

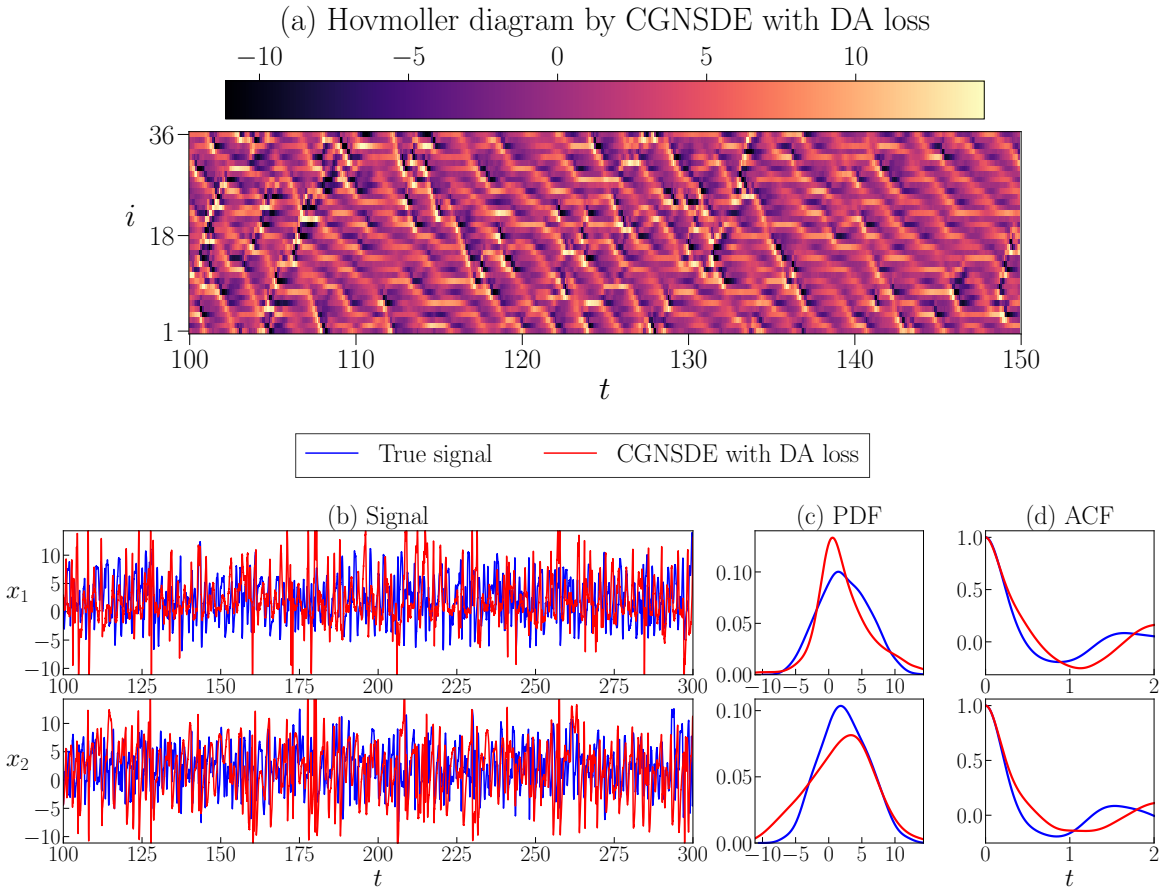

Figure 4.10: Long-term simulation results of the CGNSDE with the DA loss for the Lorenz 96 system (Case 2). Panel (a): Hovmoller diagram of all states. Panel (b): time series of the first three states. Panel (c) and (d): The PDFs and the ACFs.

Case 3: Non-CGNS that is spatially inhomogeneous

The last case involves a spatially inhomogeneous case in the true system, which does not satisfy the CGNS. The two neural networks have 5 layers with 543 parameters and 3 layers with 119 parameters, respectively.

Table 4.7 summarizes the model performance in the test period. The two CGNSDE models can improve the performance of state forecast and DA compared with the knowledge-based

regression model. The CGNSDE with the DA loss in training has the lowest DA error for the test data, with some trade-offs in forecasting accuracy compared to the CGNSDE without DA loss. The DA MSE and DA negative log-likelihood of applying EnKBF to this true inhomogeneous system are 0.0672 and 8.8427, respectively.

Table 4.7: Lorenz 96 system (Case 3): Performance of the models in the test period.

	Forecast MSE	DA MSE	DA Neg-Log-Likelihood
Knowledge-based regression model	1.1625	1.2634	58.0666
CGNSDE without DA loss	0.2984	0.4903	38.1015
CGNSDE with the DA loss	0.3752	0.3794	26.2498

Panel (a) of Figure 4.11 displays the inhomogeneous spatiotemporal pattern of the true system. Panel (b) shows DA results using the CGNSDE trained with the DA loss for several states (x_2, x_{12}, x_{24} and x_{36}), whose indices are distributed across the location range of the system. The DA results are overall accurate. Note that half of the grid points with the odd/even index numbers still share the same neural networks as in (4.27). Such a computationally efficient strategy remains working well even for this inhomogeneous case.

4.1.5 Discussion and Conclusion

In this paper, a new knowledge-based and machine-learning hybrid modeling approach, the CGNSDE, is developed. It aims to facilitate modeling complex dynamical systems and implementing the associated DA. In contrast to the standard neural network predictive models, the CGNSDE is designed to effectively tackle both forward prediction tasks and inverse state estimation problems. The unique features of the CGNSDE are summarized as follows:

1. The CGNSDE exploits a computationally robust causal inference method to develop a parsimonious model with knowledge-based components that correspond to interpretable physics.

2. Neural networks are supplemented to such knowledge-based components in a systematic way that allows closed analytic formulae for efficient DA.

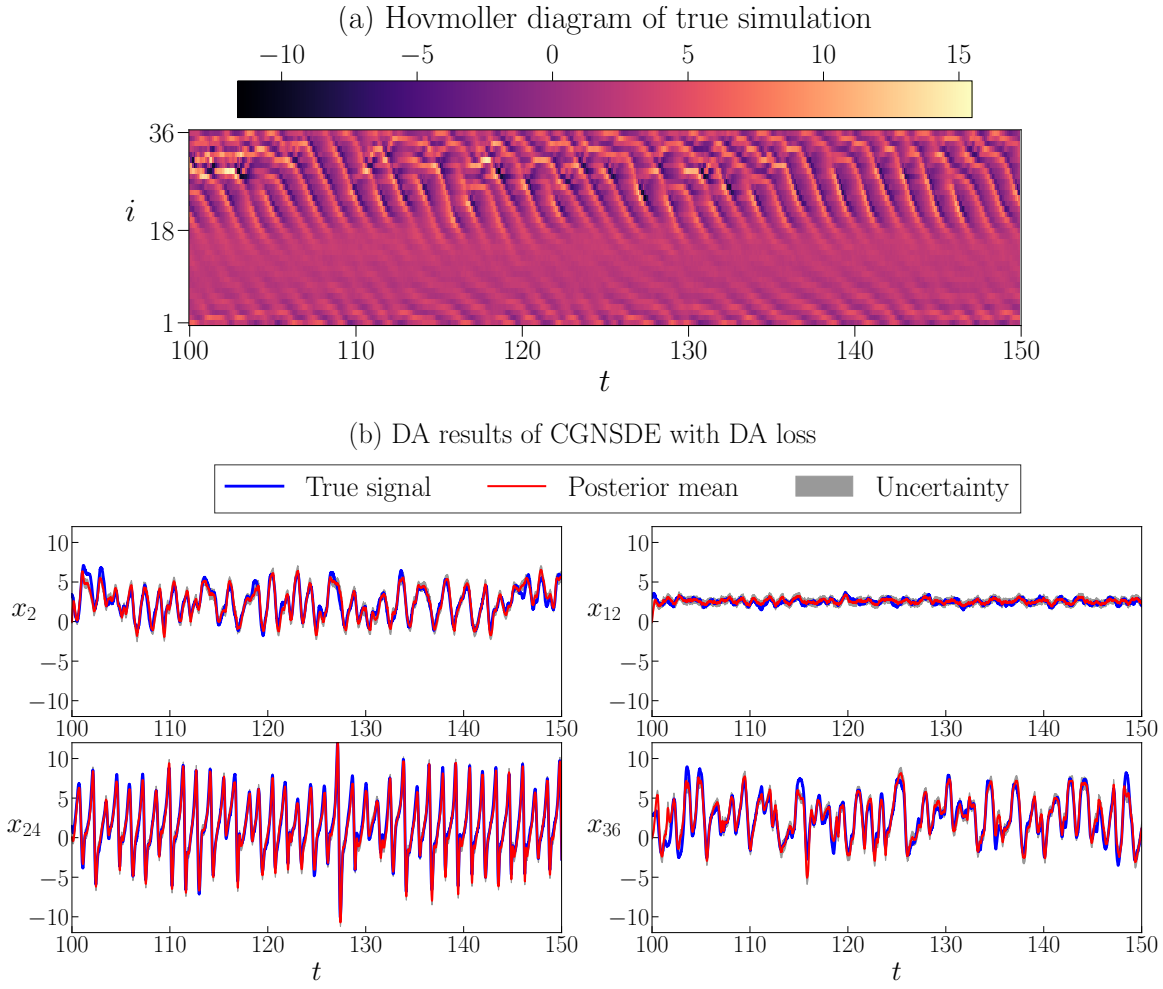

Figure 4.11: Model simulation and the DA results of the inhomogeneous Lorenz 96 system (Case 3). Panel (a): Hovmoller diagram of all states in the true system. Panel (b): DA results of the unobserved states x_2, x_{12}, x_{24}, and x_{36} using the CGNSDE with the DA loss. The associated uncertainties are indicated by the grey colored regions, which correspond to two standard deviation from the posterior mean.

3. The analytic formulae are used in an additional computationally affordable loss to train the neural networks that explicitly improve the accuracy of the CGNSDE for DA.

4. The DA loss function also advances the neural networks to enhance the performance of identifying the causal relationship of the underlying system, further strengthening the modeling skills.

5. The CGNSDE is more capable of estimating extreme events and quantifying the associated uncertainty.

6. Crucial physical properties in many complex systems, such as the translate-invariant local dependence of state variables, can significantly simplify the neural network structures and facilitate the CGNSDE to be applied to high-dimensional systems.

Utilizing a hierarchy of numerical experiments, it is shown that the CGNSDE is skillful in modeling and estimating the state of chaotic systems with intermittency and strong non-Gaussian features. In particular, the additional neural network components allow the CGNSDE to significantly outperform the standard regression models. It is also demonstrated that the DA loss in the CGNSDE plays a vital role in enhancing the skill of state estimation, which is not always highlighted in state-of-the-art neural network predictive models.

There are quite a few extensions of the current CGNSDE framework, which provides the potential for the approach to be further improved and applied to a wide range of challenging problems.

Other forms of the DA loss function. One of the crucial features of the CGNSDE is the additional DA loss. In this paper, the loss is designed in the simplest possible way by minimizing the error in the posterior mean time series related to the truth. Since the posterior mean depends on the posterior covariance in (4.2), the information in the entire posterior distribution is still utilized in such a setup. Nevertheless, the closed analytic formulae for both the mean and covariance can be exploited more delicately. One natural idea is to use the likelihood function as the DA loss (4.14), as was utilized as the validation criterion in the numerical experiment section. The likelihood can be regarded as a weighted MSE, where the square root of the time-varying covariance serves as the weight at each time instant. One obvious advantage of using likelihood loss is that uncertainty will be characterized more rigorously, especially when quantifying uncertainty and intermittency.

CGNSDE with recurrent neural network. The neural networks in all the numerical experiments of this study are the standard feed-forward networks. Such a simple setup allows us to verify the crucial role of each unique feature of the CGNSDE in improving modeling and DA skills. In practice, many other types of neural networks can be incorporated into the CGNSDE. Particularly, including the memory in the signals is helpful for prediction and state estimation. Since the DA formulae in (4.2) can depend on the entire history of the observed variables $\mathbf{u}_1(s \leqslant t)$, it justifies the incorporation of the memory effects into the neural networks in (4.4). One of the most natural ways in this direction is to adopt recurrent neural networks [113, 338] in the CGNSDE. Other advanced techniques, such as the transformer [339, 340], can also be included in the CGNSDE.

Discrete-in-time extension. Another natural extension of the current framework is considering the situation with discrete-in-time observations, which appear in many geophysical applications. This is an analog to the continuous-in-time DA framework in this paper (4.2). Closed analytic formulae still exist when the observations arrive at discrete instants [312].

4.2 Continuous-Time Conditional Gaussian Koopman Network

Complex dynamical systems are ubiquitous across a wide range of scientific and engineering fields, including climate science, geophysics, materials science, and neuroscience [1–4]. These systems are typically characterized by multiscale interactions, chaotic or turbulent dynamics, non-Gaussian distributions, intermittency, and extreme events [7–12, 275]. For decades, modeling complex dynamical systems has primarily relied on governing equations derived from physical laws [282–285], which require a deep understanding of the underlying physics. However, with the rapid growth of available data and computing resources, data-driven approaches have become increasingly promising for modeling these systems [29, 30, 109]. These methods include sparse model identification [16, 20, 21, 105], reduced-order models [17–19, 136, 287, 341], physics-informed machine learning [31–33, 39, 342–346], stochastic parameterizations [13, 66, 83, 84], and deep learning techniques [40–42, 51]. Among these, deep learning is particularly flexible in capturing nonlinear dynamics and can deliver superior forecast performance when sufficient training data is available, especially

for systems with unknown governing equations. However, the highly nonlinear, black-box nature of deep learning models poses significant challenges for outer-loop scientific machine learning applications such as data assimilation.

Data assimilation (DA) [62–65, 71, 72] integrates observational data into models to enhance state estimation. DA was originally designed to improve the estimation of initial conditions for the real-time forecast. Nowadays, DA has wide applications in parameter estimation, model identification, optimal control, and reconstructing missing data. DA consists of a two-step procedure: a model forecast generates a statistical prediction, which is then refined in the analysis step by incorporating new observations, improving accuracy, and reducing uncertainty. Given the highly nonlinear nature of many complex dynamical systems, ensemble DA [73–75, 347] has become one of the most widely used methods in practice. Ensemble DA methods utilize an ensemble of realizations to approximate probabilistic distributions of target states and update the ensemble accordingly when new observations become available. When the governing equations of dynamical systems are unknown or too costly to simulate, machine learning is often employed to assist DA. First, machine learning can be used to build data-driven forecast models for DA [87, 89, 90, 218, 259, 301–303]. Second, machine learning with observers enhances the prediction of complex systems, and the physics-informed machine learning can be used to assist in the design of observers [296, 348, 349]. Third, machine learning can streamline the entire DA process by developing end-to-end learning frameworks [81, 94, 95, 307–309]. A comprehensive review of machine learning with data assimilation can be found in [88].

While separate end-to-end machine learning models can be developed for prediction and DA, it is practically useful to design a unified deep learning framework that simultaneously tackles both tasks. The framework aims to incorporate specific dynamical structures to explicitly capture some aspects of the underlying physics, distinguishing it from fully black-box models. The unique model structures can also be used to develop efficient forecasts and DA algorithms.

In this paper, we develop a Conditional Gaussian Koopman Network (CGKN), which is a deep learning framework designed to address the challenges of modeling complex dynamical systems, performing efficient forecasts, and accelerating effective DA. The CGKN is structured as a neural stochastic differential equation. It operates in three main steps: (i) transforming the state of the original nonlinear system into that of a conditional Gaussian nonlinear system, (ii) learning the dynamics of the conditional Gaussian nonlinear system

to enable rapid state forecast and DA, and (iii) transforming the forecast and DA results back into the original system. Notably, the transformations required in the first and third steps are often unknown *a priori*, but they can be jointly learned along with the dynamics of the conditional Gaussian nonlinear system.

It is essential to highlight two crucial building blocks of the CGKN. First, the conditional Gaussian nonlinear system [193, 194, 312] encompasses a rich class of nonlinear and non-Gaussian stochastic differential equations (SDEs), embedding a conditional linear structure in the model. Many well-known complex nonlinear dynamical systems fit within the framework of conditional Gaussian nonlinear systems, which have been exploited to develop realistic systems and assist in creating numerous fast computational algorithms. Second, in contrast to standard Koopman theory, which transforms the original nonlinear system into a linear system [350–357], the CGKN aims to transform the original system into a specific nonlinear system characterized by conditional linear structures. In this context, the dynamics of the unobserved states become linear when the observed state is known, while the interactions among different variables remain highly nonlinear. This intrinsic nonlinear dynamics enables the model to capture extreme events and strong non-Gaussian features in both joint and marginal distributions, with appropriate uncertainty quantification. Notably, the conditional linear structure allows the development of analytic formulae for DA to estimate unobserved states given the time series data of the observed states. With these closed analytic formulae, the quantification of the DA performance can be easily integrated into the loss function. It serves as an additional training criterion for the deep learning framework that is both efficient and robust. The analytic formulae significantly enhance the speed of DA and eliminate the need for empirical tunings as in the ensemble methods. Additionally, it reduces the randomness and inaccuracies stemming from inevitable large sampling errors. This approach has recently been explored in [110] and holds the potential to enhance not only DA but also the quality of the learned model, particularly regarding the probabilistic relationship between observed and unobserved states.

4.2.1 Problem Statement

Considering a complex dynamical system in the general form:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathcal{M}(\mathbf{u}), \tag{4.28}$$

where $\mathbf{u} \in \mathbb{R}^{d_{\mathbf{u}}}$ denotes the system state and $\mathcal{M} : \mathbb{R}^{d_{\mathbf{u}}} \mapsto \mathbb{R}^{d_{\mathbf{u}}}$ is a real vector-valued analytic function that describes the system dynamics. The system state $\mathbf{u}$ can be a high-dimensional vector and the dynamics function $\mathcal{M}$ can be highly nonlinear. The system in the form of (4.28) can encompass a wide range of complex dynamical behaviors including multi-scale dynamics, chaos, turbulence, stochasticity, non-Gaussian statistics, intermittency, and extreme events. In this work, we focus on partially observed dynamical systems, with observed state variables denoted as $\mathbf{u}_1 \in \mathbb{R}^{d_{\mathbf{u}_1}}$ and the unobserved state variables denoted as $\mathbf{u}_2 \in \mathbb{R}^{d_{\mathbf{u}_2}}$. The unobserved states are the states that are not directly measured and to be inferred from data of observed states through techniques of data assimilation (DA). To explicitly discriminate the observed state variables $\mathbf{u}_1$ and unobserved state variables $\mathbf{u}_2$, the true system in (4.28) is rewritten as:

$$\begin{aligned} \frac{d\mathbf{u}_1}{dt} &= \mathcal{M}_1(\mathbf{u}_1, \mathbf{u}_2), \\ \frac{d\mathbf{u}_2}{dt} &= \mathcal{M}_2(\mathbf{u}_1, \mathbf{u}_2), \end{aligned} \tag{4.29}$$

where $[\mathbf{u}_1; \mathbf{u}_2] = \mathbf{u}$, $\mathcal{M}_1 : \mathbb{R}^{d_{\mathbf{u}}} \mapsto \mathbb{R}^{d_{\mathbf{u}_1}}$ and $\mathcal{M}_2 : \mathbb{R}^{d_{\mathbf{u}}} \mapsto \mathbb{R}^{d_{\mathbf{u}_2}}$ are real vector-valued analytic functions. The notation $[\cdot;\cdot]$ is used to denote the vertical concatenation of two column vectors. The DA task targets at the conditional distribution $p(\mathbf{u}_2(t)|\mathbf{u}_1(s), s \leqslant t)$, where the trajectory of $\mathbf{u}_1$ up to time t, namely $\mathbf{u}_1(s \leqslant t)$, is the observations while the state of $\mathbf{u}_2$ at t needs to be estimated [63–65].

As a fundamental component of deep learning, a neural network is a computational model inspired by the structure of biological neural networks, consisting of hierarchical layers of neurons, each computing a weighted sum of inputs followed by a non-linear activation function. The universal approximation theorem [26, 34, 35] states that a neural network with at least one hidden layer, a sufficient number of neurons, and a non-linear activation function can approximate any continuous function. Therefore, both $\mathcal{M}_1$ and $\mathcal{M}_2$, which can be highly nonlinear functions, can be approximated by neural networks according to universal approximation theory. With a sufficient amount of data, the approximation of neural networks can be quite accurate, leading to a good performance of state forecasts. However, the DA for the black-box neural-networks-based model would require techniques such as the ensemble-based method, which are computationally expensive, could suffer from sampling errors, and often require empirical tuning.

In this work, we develop the Conditional Gaussian Koopman Network (CGKN) for both state forecast and efficient DA. Instead of directly building a data-driven model for state variables $\mathbf{u}_1$ and $\mathbf{u}_2$, the proposed framework leverages a generalized version of the Koopman theory and exploits a proper transformation to map the unobserved state variables $\mathbf{u}_2$ to latent state variables $\mathbf{v}$, such that those latent variables are conditionally linear in the system dynamics. Therefore, the observed state variables $\mathbf{u}_1$ and the latent state variables $\mathbf{v}$ can be modeled by a conditional Gaussian nonlinear system [193, 194, 312], which encompasses many nonlinear models across various disciplines with many applications in natural science and engineering [257, 268, 315, 320–322, 358]. Importantly, the conditional Gaussian nonlinear system facilitates the development of closed analytic formulae for DA. The analytic formulae provide more efficient DA than those techniques developed for nonlinear models and also enable the incorporation of DA performance into the loss function when training the CGKN model. It is essential to highlight that the governing equations of complex dynamical systems do not need to be known as CGKN is a data-driven method based on deep learning. If the governing equations are available, CGKN can be learned from the synthetic data generated from the simulation of the complex system and then provides efficient data assimilation. Alternatively, even if the governing equations are unknown, CGKN can still be learned using real data, for which this work assumes the data of unobserved states is accessible at the offline training stage. With a pre-trained CGKN, the inference processes, including both state prediction and efficient DA, do not require data from unobserved states. A schematic overview of the CGKN framework, including its application in state forecast and DA, is presented in Figure 4.12.

4.2.2 Generalized Koopman Operator for Continuous-Time System

Koopman theory [350] provides a linear perspective on nonlinear dynamical systems in the form of (4.28) by describing the system in terms of the evolution of observable of the system state. The observable $h : \mathbb{R}^{d_{\mathbf{u}}} \mapsto \mathbb{R}$ is a function in the Hilbert space operating on the system state $\mathbf{u}$. For the complex dynamical system in (4.28), the state transition operator $\mathcal{G}^{\Delta t} : \mathbb{R}^{d_{\mathbf{u}}} \mapsto \mathbb{R}^{d_{\mathbf{u}}}$ over time interval Δt is defined as:

$$\mathcal{G}^{\Delta t}\big(\mathbf{u}(t)\big) := \mathbf{u}(t) + \int_t^{t+\Delta t} \mathcal{M}\big(\mathbf{u}(\tau)\big)\, d\tau. \tag{4.30}$$

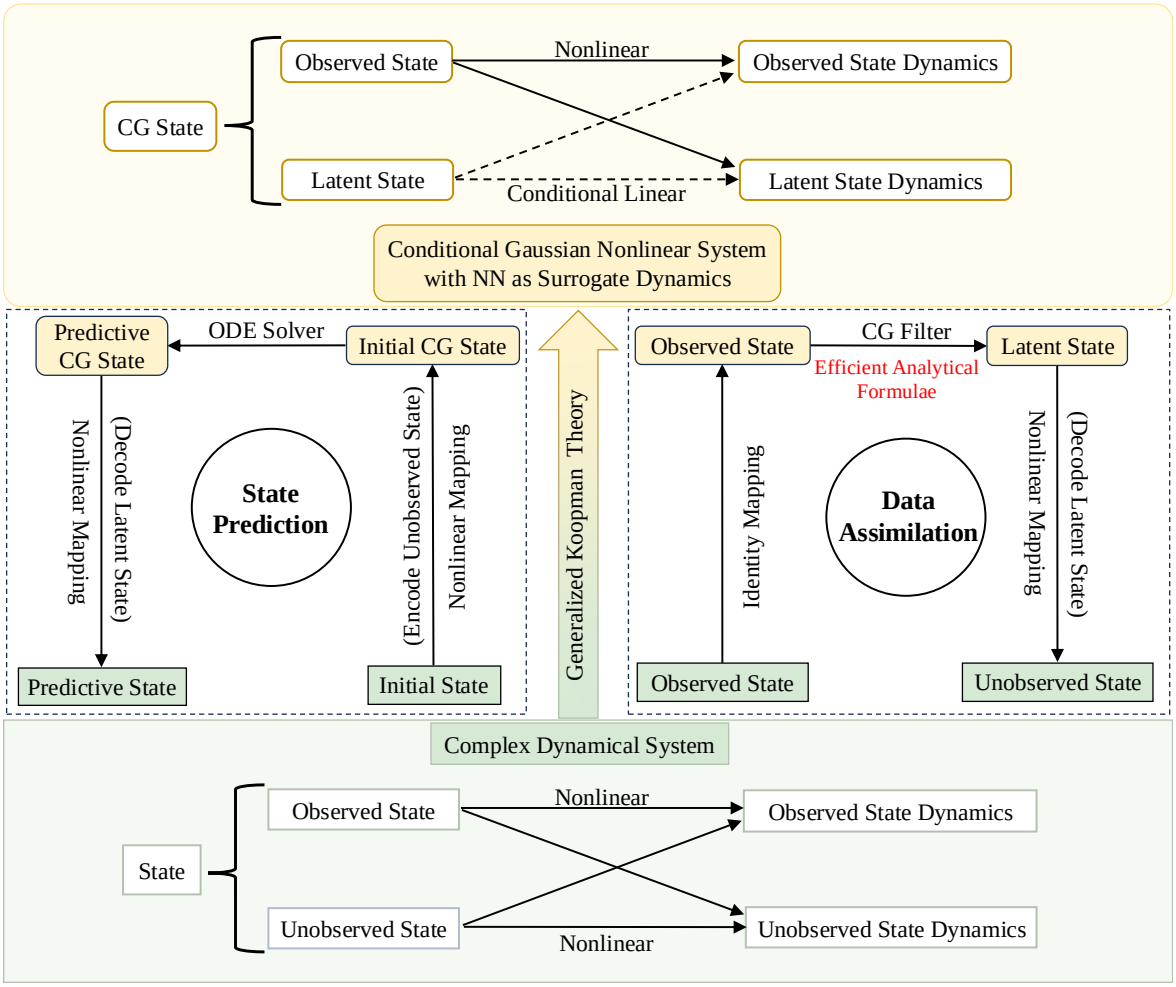

Figure 4.12: Schematic diagram of conditional Gaussian Koopman network (CGKN) for a partially observed system. The complex nonlinear dynamical system is transformed into the conditional Gaussian nonlinear system via a generalized usage of Koopman theory, which maps between the unobserved state variables and the latent state variables featured by conditional linear dynamics. The unknown dynamics of the conditional Gaussian nonlinear system are constructed by neural networks and jointly learned with the unknown nonlinear mappings. The analytic DA formulae for the conditional Gaussian nonlinear system can significantly accelerate DA process and also allow a computationally affordable DA loss to be incorporated into the CGKN training process. The trained CGKN model can be used for state forecasts and efficient DA, for which the computations are mainly performed in the conditional Gaussian state space and then mapped back to the original state space.

The Koopman operator $\mathcal{K}^{\Delta t}$ is a linear operator acting on observable function h which is defined as

$$\mathcal{K}^{\Delta t}(h) := h \circ \mathcal{G}^{\Delta t}, \tag{4.31}$$

where $\circ$ denotes function composition.

Furthermore, the Koopman infinitesimal generator $\mathcal{L}$ is defined as:

$$\mathcal{L} := \lim_{\Delta t \to 0} \frac{\mathcal{K}^{\Delta t}(h) - h}{\Delta t}, \tag{4.32}$$

which characterizes the $\mathcal{L}$ as the rate of change of the observable h since $\mathcal{K}^{\Delta t}(h(\mathbf{u}(t))) = h \circ \mathcal{G}^{\Delta t}(\mathbf{u}(t)) = h(\mathbf{u}(t + \Delta t))$.

Therefore, the evolution of observable h is governed by the Koopman infinitesimal generator $\mathcal{L}$ which is a linear operator:

$$\frac{\partial h}{\partial t} = \mathcal{L} h. \tag{4.33}$$

In real application, the infinite-dimensional linear dynamical system and the linear operator $\mathcal{L}$ are approximated by the finite-dimensional representations:

$$\frac{d\mathbf{v}}{dt} = \mathbf{A}\mathbf{v}, \tag{4.34}$$

where $\mathbf{v} \in \mathbb{R}^{d_v}$ is a finite-dimensional vector that approximates the infinite-dimensional function h and the matrix $\mathbf{A} \in \mathbb{R}^{d_v \times d_v}$ is the linear dynamics of this discrete linear dynamical system which approximates the linear operator $\mathcal{L}$.

In this work, a generalized application of Koopman theory has been employed to linearize the unobserved states $\mathbf{u}_2$ while retaining the nonlinearity of observed states $\mathbf{u}_1$ for the partially observed dynamical systems in (4.29). The resulting conditional linear dynamical system is called the Conditional Gaussian Nonlinear System (CGNS), which is written as:

$$\begin{aligned} \frac{d\mathbf{u}_1}{dt} &= \mathbf{f}_1(\mathbf{u}_1) + \mathbf{g}_1(\mathbf{u}_1)\mathbf{v} + \boldsymbol{\sigma}_1 \dot{\mathbf{W}}_1, \\ \frac{d\mathbf{v}}{dt} &= \mathbf{f}_2(\mathbf{u}_1) + \mathbf{g}_2(\mathbf{u}_1)\mathbf{v} + \boldsymbol{\sigma}_2 \dot{\mathbf{W}}_2, \end{aligned} \tag{4.35}$$

where $\mathbf{f}_1 : \mathbb{R}^{d_{u_1}} \mapsto \mathbb{R}^{d_{u_1}}$, $\mathbf{g}_1 : \mathbb{R}^{d_{u_1}} \mapsto \mathbb{R}^{d_{u_1} \times d_v}$, $\mathbf{f}_2 : \mathbb{R}^{d_{u_1}} \mapsto \mathbb{R}^{d_v}$, and $\mathbf{g}_2 : \mathbb{R}^{d_{u_1}} \mapsto \mathbb{R}^{d_v \times d_v}$

are nonlinear functions of $\mathbf{u}_1$ to be learned from data. $\mathbf{W}_1 \in \mathbb{R}^{d_{\mathbf{u}_1}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{\mathbf{v}}}$ are independent Wiener processes with $\boldsymbol{\sigma}_1 \in \mathbb{R}^{d_{\mathbf{u}_1} \times d_{\mathbf{u}_1}}$ and $\boldsymbol{\sigma}_2 \in \mathbb{R}^{d_{\mathbf{v}} \times d_{\mathbf{v}}}$ as the noise coefficients. $\mathbf{v} = \boldsymbol{\varphi}(\mathbf{u}_2) \in \mathbb{R}^{d_{\mathbf{v}}}$ is called latent states with a nonlinear encoder $\boldsymbol{\varphi} : \mathbb{R}^{d_{\mathbf{u}_2}} \mapsto \mathbb{R}^{d_{\mathbf{v}}}$. It is worth noting that the conditional distribution of the latent variables $p(\mathbf{v}(t)|\mathbf{u}_1(s), s \leqslant t)$ is Gaussian, due to the conditional linear structure in (4.35), which facilitates efficient analytic formulae of DA with more details discussed in Section 4.2.3.2. With the state prediction and DA results of $[\mathbf{u}_1; \mathbf{v}]$, the corresponding original unobserved state variables can be obtained via $\mathbf{u}_2 = \boldsymbol{\psi}(\mathbf{v})$ with the nonlinear decoder $\boldsymbol{\psi} : \mathbb{R}^{d_{\mathbf{v}}} \mapsto \mathbb{R}^{d_{\mathbf{u}_2}}$. The temporal derivative of the system states $d\mathbf{u}/dt$ is sometimes denoted as $\dot{\mathbf{u}}$ for short. The encoder $\boldsymbol{\varphi}$, decoder $\boldsymbol{\psi}$, and the four nonlinear functions $\mathbf{f}_1$, $\mathbf{g}_1$, $\mathbf{f}_2$, and $\mathbf{g}_2$ are parameterized by neural networks and are jointly learned from data $\{\mathbf{u}^\star(t_n)\}_{n=0}^N$. The neural networks that approximate the four nonlinear functions are collectively referred to as $\boldsymbol{\eta}$.

4.2.3 Architecture of Continuous CGKN

The CGKN comprises an encoder $\boldsymbol{\varphi}$, a decoder $\boldsymbol{\psi}$, and sub-networks $\boldsymbol{\eta}$. More specifically, the architecture of CGKN consists of (i) an encoder that maps the original unobserved system state variables to the latent state variables of a conditional Gaussian nonlinear system, (ii) neural networks that approximate the unknown terms in the dynamics of the conditional Gaussian nonlinear system, and (iii) a decoder that transforms the latent state variables of the conditional Gaussian nonlinear system back to the original unobserved system state variables. The architecture of CGKN is illustrated in Figure 4.13(a). As a high-level deep learning framework, CGKN can integrate various neural network architectures, including fully-connected neural networks, residual neural networks, convolutional neural networks, and recurrent neural networks for encoder $\boldsymbol{\varphi}$, decoder $\boldsymbol{\psi}$, and sub-nets $\boldsymbol{\eta}$. In this work, the fully-connected neural networks are used to demonstrate the effectiveness and capability of CGKN.

It is worth noting that the CGKN framework is also applicable to another general type of a partially observed system, which consists of a dynamical model for the whole system state and an observation mapping that partially extracts observable information from the system state. This type of partially observed system can be formalized in the same form as shown in (4.29) and then modeled by the CGKN in (4.35). This is achieved by retaining the dynamical model and modeling the dynamics of the observations in relation to the system

states, where the system state corresponds to the unobserved states $\mathbf{u}_2$ and the observations correspond to the observed states $\mathbf{u}_1$ in (4.35). If the dynamics of observations do not exist or if the observation time interval is large, a discrete-time version of CGKN can be employed.

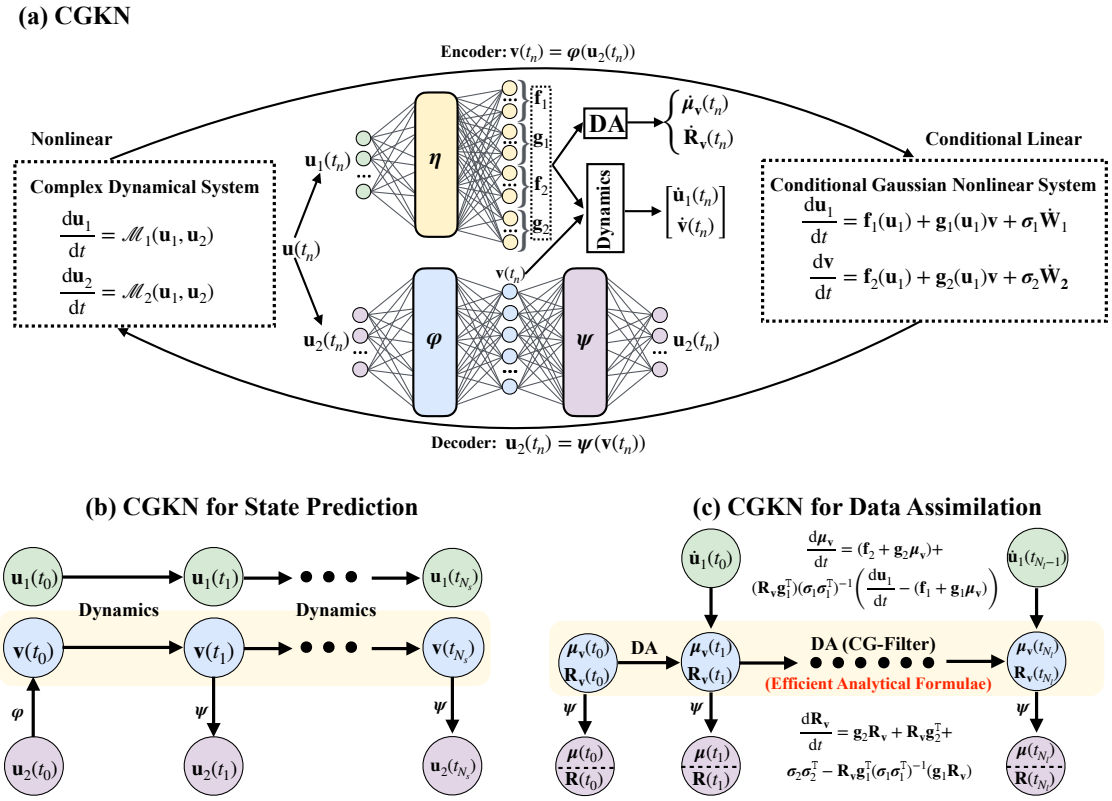

Figure 4.13: The architecture of CGKN and its applications for state prediction and data assimilation. The CGKN approximates the true nonlinear complex dynamical system by a modeled system with a conditional linear structure, which is also known as a conditional Gaussian nonlinear system. The conditional linear structure is enabled by a proper choice of latent state variables $\mathbf{v}$ as illustrated in (a). The nonlinear mappings between the original state variables $\mathbf{u}_2$ and the latent state variables $\mathbf{v}$ are achieved via the encoder $\boldsymbol{\varphi}$ and decoder $\boldsymbol{\psi}$, which are jointly learned with the unknown functions $\mathbf{f}_1$, $\mathbf{g}_1$, $\mathbf{f}_2$, $\mathbf{g}_2$ of the conditional Gaussian nonlinear system. The pre-trained CGKN can be used for state prediction and efficient data assimilation as illustrated in (b) and (c).

4.2.3.1 Continuous CGKN for State Forecast

The CGKN model can be used for state prediction given an initial state $\mathbf{u}^\star(\mathrm{t}_0) = [\mathbf{u}_1^\star(\mathrm{t}_0); \mathbf{u}_2^\star(\mathrm{t}_0)]$. Firstly, the unobserved state variables $\mathbf{u}_2^\star(\mathrm{t}_0)$ are transformed into la-

tent state variables $\mathbf{v}^\star(t_0)$ via an encoder $\boldsymbol{\varphi}$, which leads to the initial state of conditional Gaussian nonlinear system $\mathbf{u}_{CG} = [\mathbf{u}_1; \mathbf{v}]$ for (4.35):

$$\mathbf{u}^\star_{CG}(t_0) = \begin{bmatrix} \mathbf{u}_1^\star(t_0) \\ \mathbf{v}^\star(t_0) \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1^\star(t_0) \\ \boldsymbol{\varphi}(\mathbf{u}_2^\star(t_0)) \end{bmatrix}. \tag{4.36}$$

Then, the state prediction of the conditional Gaussian nonlinear system at a future time t_n can be obtained by:

$$\mathbf{u}_{CG}(t_n) = \begin{bmatrix} \mathbf{u}_1(t_n) \\ \mathbf{v}(t_n) \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1^\star(t_0) + \int_{t_0}^{t_n} (\mathbf{f}_1(\mathbf{u}_1(t)) + \mathbf{g}_1(\mathbf{u}_1(t))\mathbf{v}) dt \\ \mathbf{v}^\star(t_0) + \int_{t_0}^{t_n} (\mathbf{f}_2(\mathbf{u}_1(t)) + \mathbf{g}_2(\mathbf{u}_1(t))\mathbf{v}) dt \end{bmatrix}. \tag{4.37}$$

Lastly, the predictive latent state variables $\mathbf{v}(t_n)$ are transformed back into the original unobserved state variables $\mathbf{u}_2(t_n)$ via a decoder $\boldsymbol{\psi}$:

$$\mathbf{u}(t_n) = \begin{bmatrix} \mathbf{u}_1(t_n) \\ \mathbf{u}_2(t_n) \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1(t_n) \\ \boldsymbol{\psi}(\mathbf{v}(t_n)) \end{bmatrix}. \tag{4.38}$$

In this work, the encoder $\boldsymbol{\varphi}$, the decoder $\boldsymbol{\psi}$, and the functions $\mathbf{f}_1$, $\mathbf{g}_1$, $\mathbf{f}_2$, $\mathbf{g}_2$ are jointly learned to ensure a good state prediction performance, i.e., the difference between the state prediction $\mathbf{u}(t_n)$ and the true system state $\mathbf{u}^\star(t_n)$ is small for a series of t_n. This is achieved by incorporating the state prediction performance into the loss function, and more details of the loss function for training a CGKN model are summarized in Section 4.2.4. An illustration of CGKN for state prediction is shown in Figure 4.13(b).

If the initial value of unobserved states $\mathbf{u}_2^\star(t_0)$ are not given, the estimated unobserved states via data assimilation after a warm-up period can substitute for the unknown true initial value. The CGKN for DA is introduced in Section 4.2.3.2. Forecasting the entire system state using data from partially observed states is one of the features of CGKN, and it is worth noting that the performance of the state forecast will thus depend on the quality of DA estimated unobserved states.

4.2.3.2 Continuous CGKN for Data Assimilation

In addition to the state prediction, the CGKN model can also be used for efficient DA. This is achieved by the conditional linear structure in (4.35) and accounts for one of the key highlights in the proposed CGKN framework. More specifically, despite the highly

nonlinear dynamics of the coupled system, the conditional distribution $p(\mathbf{v}(t)|\mathbf{u}_1(s), s \leqslant t) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{v}}(t)), \mathbf{R}_{\mathbf{v}}(t))$ is Gaussian. The mean $\boldsymbol{\mu}_{\mathbf{v}}$ and covariance $\mathbf{R}_{\mathbf{v}}$ of this conditional Gaussian distribution can be solved by closed analytic formulae:

$$\begin{aligned} \frac{d\boldsymbol{\mu}_{\mathbf{v}}}{dt} &= (\mathbf{f}_2 + \mathbf{g}_2\boldsymbol{\mu}_{\mathbf{v}}) + (\mathbf{R}_{\mathbf{v}}\mathbf{g}_1^T)(\boldsymbol{\sigma}_1\boldsymbol{\sigma}_1^T)^{-1}\left(\frac{d\mathbf{u}_1}{dt} - (\mathbf{f}_1 + \mathbf{g}_1\boldsymbol{\mu}_{\mathbf{v}})\right), \\ \frac{d\mathbf{R}_{\mathbf{v}}}{dt} &= \mathbf{g}_2\mathbf{R}_{\mathbf{v}} + \mathbf{R}_{\mathbf{v}}\mathbf{g}_2^T + \boldsymbol{\sigma}_2\boldsymbol{\sigma}_2^T - \mathbf{R}_{\mathbf{v}}\mathbf{g}_1^T(\boldsymbol{\sigma}_1\boldsymbol{\sigma}_1^T)^{-1}(\mathbf{g}_1\mathbf{R}_{\mathbf{v}}). \end{aligned} \tag{4.39}$$

Given the data of observed state variables $\{\mathbf{u}_1^\star(t_n)\}_{n=0}^N$, the posterior mean of latent state variables $\{\boldsymbol{\mu}_{\mathbf{v}}(t_n)\}_{n=0}^N$ in (4.35) can be estimated based on the analytic formulae in (4.39). Similar to the latent state variables, the posterior mean $\boldsymbol{\mu}_{\mathbf{v}}$ of the latent state variables can also be transformed back to the mean estimation $\boldsymbol{\mu}$ of the original unobserved state variables $\mathbf{u}_2$ via the decoder ψ, i.e., $\boldsymbol{\mu}(t_n) \approx \psi(\boldsymbol{\mu}_{\mathbf{v}}(t_n))$. Although this direct nonlinear transformation of the mean values is efficient, it is generally not valid for an arbitrary nonlinear decoder. However, its validity can be greatly enhanced by a trainable decoder with the DA loss being considered. In the DA process, the exact initial condition or initial distribution of the DA solution is usually unknown. As a result, it takes some time for the DA solution to stabilize and adjust to eliminate the inconsistency from the initialization. During this initial period of DA, which is known as the warm-up period, the estimated states may deviate from the actual values. Consequently, warm-up period is crucial for ensuring that DA produces stable and accurate estimates for unobserved states before proceeding with any analysis or decision-making. Therefore, the steps of DA for the warm-up period are omitted in the evaluation of DA loss for training the CGKN. More details about the DA loss are discussed in Section 4.2.4. An illustration of CGKN for DA is shown in Figure 4.13(c).

The effectiveness of DA depends on the observability of unobserved states, which is the intrinsic property of the complex system described in (4.29). The observability of unobserved states is determined by the dynamics functions $\mathcal{M}_1$ and $\mathcal{M}_2$. For instance, $\mathbf{u}_2$ is fully observable if the observed states $\mathbf{u}_1$ sufficiently dictate the dynamics of $\mathbf{u}_2$ in such a way that observing $\mathbf{u}_1$ over time gives all the necessary information to estimate $\mathbf{u}_2$ without ambiguity. If the $\mathcal{M}_1$ and $\mathcal{M}_2$ are linear functions, the observability matrix condition can be used to determine the full observability. $\mathbf{u}_2$ is not observable if the dynamics function $\mathcal{M}_1$ of the observed state does not include $\mathbf{u}_2$. In this case, the observation of $\mathbf{u}_1$ does not provide any information to infer the unobserved state $\mathbf{u}_2$, and consequently, CGKN will be

unable to accurately predict future states using the estimated unobserved states from DA.

Efficient DA can be performed by CGKN with the analytic formulae in (4.39). The computational complexity for this analytic approach is $\mathcal{O}(N^3)$ where $N = d_{\mathbf{v}}$ is the dimension of the latent state. As a comparison, the computational complexity of ensemble DA methods is $\mathcal{O}(JM^2)$, where J is the ensemble size and $M = d_{\mathbf{u}_2}$ is the dimension of unobserved states. To accurately approximate the probabilistic distribution of an N-dimensional random vector, an ensemble size $J = c^M$ is typically required, where c denotes the sample size for a single random variable. This leads to a resulting complexity of $\mathcal{O}(c^M M^2)$, which grows exponentially due to the curse of dimensionality. In real-world applications, a much smaller ensemble size is often adopted due to the restriction of computational resources, which inevitably leads to issues originating from sampling errors, and various techniques (e.g., localization) have been developed to empirically deal with insufficient sample size in a high-dimensional space. Therefore, compared with ensemble-based DA methods, which are commonly used for nonlinear models, the analytic DA formulae of the CGKN model can avoid sampling errors and reduce the computational cost. This allows the DA results of CGKN to be incorporated into the loss function during the training procedure. In each training step, the difference between the data of true unobserved state $\mathbf{u}_2^\star$ and the posterior mean $\boldsymbol{\mu}$ is quantified by a DA loss term, which promotes the DA performance of the trained CGKN model. More details about the DA loss term and total loss function are summarized in Section 4.2.4.

4.2.4 Learning Continuous CGKN from Data

In this work, the proposed CGKN model involves several terms to be determined from data, including the encoder $\boldsymbol{\varphi}$, the decoder $\boldsymbol{\psi}$, and functions $\mathbf{f}_1$, $\mathbf{g}_1$, $\mathbf{f}_2$, and $\mathbf{g}_2$ in the conditional Gaussian nonlinear system of (4.35). As illustrated in Figure 4.13, these unknown terms are approximated by an autoencoder that contains both the encoder and the decoder and deep neural networks for the unknown functions in the conditional Gaussian nonlinear system. To train the autoencoder and the deep neural networks jointly, the total loss function consists of four types of loss terms.

The first loss term is the autoencoder loss L_{AE}, which essentially ensures that the latent state variables $\mathbf{v}$ contain enough information of the original unobserved state variables $\mathbf{u}_2$. This is achieved by minimizing the difference between the true data $\mathbf{u}_2^\star$ and the outputs

of autoencoder $\boldsymbol{\psi}(\mathbf{v}^\star)$, where $\mathbf{v}^\star = \boldsymbol{\varphi}(\mathbf{u}_2^\star)$. It is worth noting that the tunable nonlinearity within the encoder $\boldsymbol{\varphi}$ allows the conditional linear dynamics of the latent state variables $\mathbf{v}$ in (4.35), which is inspired by the Koopman theory, even though the true dynamics of $\mathbf{u}_2$ can be strongly nonlinear. The autoencoder loss L_{AE} is defined as $\mathbb{E}_{\mathbf{u}_2^\star}\|\mathbf{u}_2^\star - \boldsymbol{\psi}(\boldsymbol{\varphi}(\mathbf{u}_2^\star))\|^2$.

The second and third loss terms are the forecast loss of the original state denoted as $L_{\mathbf{u}}$ and the forecast loss of the latent state denoted as $L_{\mathbf{v}}$. The $L_{\mathbf{u}}$ quantifies the difference between true state $\mathbf{u}^\star$ and the state prediction $\mathbf{u}$, while the $L_{\mathbf{v}}$ measures the difference between true latent state $\mathbf{v}^\star = \boldsymbol{\varphi}(\mathbf{u}_2^\star)$ and latent state prediction $\mathbf{v}$. The motivation for incorporating the forecast loss of the original state variables $\mathbf{u}$ is straightforward, as one of the applications for CGKN model is to provide state prediction of the true complex dynamical system. On the other hand, we found that the forecast loss $L_{\mathbf{v}}$ of the latent state variables $\mathbf{v}$ can enhance the predictive performance of the CGKN model, and the main reason is that the forecast loss $L_{\mathbf{v}}$ promotes the predictive capability of the conditional Gaussian nonlinear system in (4.35). The formulas of forecast loss of the original state $L_{\mathbf{u}}$ and the forecast loss of the latent state $L_{\mathbf{v}}$ are $\mathbb{E}_{\mathbf{u}^\star(t_0)}\frac{1}{N_s}\sum_{i=1}^{N_s}\|\mathbf{u}^\star(t_i) - \mathbf{u}(t_i)\|^2$ and $\mathbb{E}_{\mathbf{u}^\star(t_0)}\frac{1}{N_s}\sum_{i=1}^{N_s}\|\mathbf{v}^\star(t_i) - \mathbf{v}(t_i)\|^2$, respectively, where N_s represents the number of forecast steps.

The fourth loss term is the DA loss L_{DA}, which is enabled by the efficient analytic formulae in (4.39) to obtain the posterior mean $\boldsymbol{\mu}_{\mathbf{v}}$ of the latent state variables $\mathbf{v}$. It is worth noting that the DA performance of the original unobserved state variables $\mathbf{u}_2$ is of actual interest. In this work, we directly map the posterior mean $\boldsymbol{\mu}_{\mathbf{v}}$ to approximate the posterior mean $\boldsymbol{\mu}$ of $\mathbf{u}_2$ via the decoder, i.e., $\boldsymbol{\mu} \approx \boldsymbol{\psi}(\boldsymbol{\mu}_{\mathbf{v}})$. Although such an approximation is often invalid for an arbitrary nonlinear decoder $\boldsymbol{\psi}$, we enforce the difference between the mapped posterior mean $\boldsymbol{\mu}$ and the data of $\mathbf{u}_2$ to be small via the DA loss term, which effectively promotes the nonlinearity within the trained decoder such that the direct mapping of posterior mean values serves as a good approximation. In addition, the DA loss can potentially enhance the discovery of probabilistic relation between the observed state variables $\mathbf{u}_1$ and the unobserved ones $\mathbf{u}_2$ when training the CGKN model. The DA loss L_{DA} is defined as $\mathbb{E}_{\mathbf{u}^\star(t_0)}\frac{1}{N_l - N_b}\sum_{j=N_b+1}^{N_l}\|\mathbf{u}_2^\star(t_j) - \boldsymbol{\mu}(t_j)\|^2$, where N_l is the number of DA steps and N_b is the warm-up period steps.

The total loss function for the training of CGKN model is then defined as:

$$L := \lambda_{AE}L_{AE} + \lambda_{\mathbf{u}}L_{\mathbf{u}} + \lambda_{\mathbf{v}}L_{\mathbf{v}} + \lambda_{DA}L_{DA}, \tag{4.40}$$

with

$$\mathrm{L}_{\mathrm{AE}} := \mathbb{E}_{\mathbf{u}_2^\star} \|\mathbf{u}_2^\star - \boldsymbol{\psi}(\boldsymbol{\varphi}(\mathbf{u}_2^\star))\|^2, \tag{4.41}$$

$$\mathrm{L}_{\mathbf{u}} := \mathbb{E}_{\mathbf{u}^\star(t_0)} \frac{1}{N_s} \sum_{i=1}^{N_s} \|\mathbf{u}^\star(t_i) - \mathbf{u}(t_i)\|^2, \tag{4.42}$$

$$\mathrm{L}_{\mathbf{v}} := \mathbb{E}_{\mathbf{u}^\star(t_0)} \frac{1}{N_s} \sum_{i=1}^{N_s} \|\mathbf{v}^\star(t_i) - \mathbf{v}(t_i)\|^2, \tag{4.43}$$

$$\mathrm{L}_{\mathrm{DA}} := \mathbb{E}_{\mathbf{u}^\star(t_0)} \frac{1}{N_l - N_b} \sum_{j=N_b+1}^{N_l} \|\mathbf{u}_2^\star(t_j) - \boldsymbol{\mu}(t_j)\|^2, \tag{4.44}$$

where $\|\cdot\|$ denotes the standard vector ℓ^2-norm and $\mathbf{u}^\star$ indicates data from the true system, and the expectations are estimated based on the samples in the training data. The above loss terms involve some hyper-parameters, including the total time steps N_s for the state forecast, the total time steps N_l for DA, and the warm-up period steps N_b for DA. For the state forecast loss terms, the CGKN model starts from the same initial condition at time t_0 as the training data from the true system. For the DA loss term, a warm-up period is often needed due to the effect of an inaccurate initial condition, and the warm-up period is excluded from the evaluation of DA loss term.

The architecture of the CGKN model has been presented in Figure 4.13(a), which consists of three neural-network-based components, including the encoder $\boldsymbol{\varphi}$, the decoder $\boldsymbol{\psi}$, and a neural network $\boldsymbol{\eta}$ to approximate the unknown functions $\mathbf{f}_1$, $\mathbf{g}_1$, $\mathbf{f}_2$, and $\mathbf{g}_2$ in the conditional Gaussian nonlinear system. Learning the unknown coefficients of those neural-network-based components from data is to seek a minimizer of the following optimization problem:

$$\min_{\boldsymbol{\theta}_{\boldsymbol{\varphi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}, \boldsymbol{\theta}_{\boldsymbol{\eta}}} \Big(\lambda_{\mathrm{AE}} \mathrm{L}_{\mathrm{AE}}(\boldsymbol{\theta}_{\boldsymbol{\varphi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}) + \lambda_{\mathbf{u}} \mathrm{L}_{\mathbf{u}}(\boldsymbol{\theta}_{\boldsymbol{\varphi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}, \boldsymbol{\theta}_{\boldsymbol{\eta}}) + \lambda_{\mathbf{v}} \mathrm{L}_{\mathbf{v}}(\boldsymbol{\theta}_{\boldsymbol{\varphi}}, \boldsymbol{\theta}_{\boldsymbol{\eta}}) + \lambda_{\mathrm{DA}} \mathrm{L}_{\mathrm{DA}}(\boldsymbol{\theta}_{\boldsymbol{\psi}}, \boldsymbol{\theta}_{\boldsymbol{\eta}}) \Big), \tag{4.45}$$

which can be solved by standard stochastic gradient descent algorithms, and the gradient is readily available via the implementation of the CGKN model on a platform that supports automatic differentiation. A common choice for the gains is $\lambda_{\mathrm{AE}} = 1/d_{\mathbf{u}_2}$, $\lambda_{\mathbf{u}} = 1/d_{\mathbf{u}}$, $\lambda_{\mathbf{v}} = 1/d_{\mathbf{u}}$, and $\lambda_{\mathrm{DA}} = 1/d_{\mathbf{u}_2}$, which scale each loss function by a factor of the inverse of the respective vector's dimension. The resulting loss functions are the mean squared error (MSE) between the true data and the approximated data. In practice, the gains can also be

set as hyper-parameters or manually adjusted based on the domain knowledge of users.

4.2.5 Uncertainty Quantification for Continuous CGKN

4.2.5.1 Uncertianty Quantification for State Forecast

It is important to note that the noise coefficients $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ of the conditional Gaussian nonlinear system in (4.35) are required to evaluate the DA loss term for the CGKN model using the analytic formulae in (4.39). The coefficient $\boldsymbol{\sigma}_1$ represents the noise associated with the observed state $\mathbf{u}_1$. Assuming $\boldsymbol{\sigma}_1$ is a diagonal matrix, its diagonal elements can be estimated through the quadratic variation of a pre-trained CGKN model without the DA loss component:

$$\begin{aligned} \mathrm{diag}(\boldsymbol{\sigma}_1) &= \sqrt{\frac{\Delta t}{N_t}\sum_{n=1}^{N_t}\big(\dot{\mathbf{u}}_1^\star(t_n)-\dot{\mathbf{u}}_1(t_n)\big)\odot\big(\dot{\mathbf{u}}_1^\star(t_n)-\dot{\mathbf{u}}_1(t_n)\big)}, \\ \mathrm{diag}(\boldsymbol{\sigma}_2) &= \sqrt{\frac{\Delta t}{N_t}\sum_{n=1}^{N_t}\big(\dot{\mathbf{v}}^\star(t_n)-\dot{\mathbf{v}}(t_n)\big)\odot\big(\dot{\mathbf{v}}^\star(t_n)-\dot{\mathbf{v}}_1(t_n)\big)}, \end{aligned} \tag{4.46}$$

where Δt denotes the time step of the training data, N_t represents the total number of time steps in the training data, and the notation $\odot$ signifies element-wise multiplication. In (4.46), $\dot{\mathbf{u}}_1^\star(t_n)$ is the time derivative of the true observed state variables, which can be computed by taking the numerical derivatives of $\mathbf{u}_1$ at time t_n, while $\dot{\mathbf{u}}_1(t_n)$ is the time derivative of the corresponding prediction from the pre-trained CGKN model without the DA loss. The $\dot{\mathbf{v}}^\star(t_n)$ is the time derivative of true latent states $\mathbf{v}^\star(t_n)=\boldsymbol{\varphi}(\mathbf{u}_2^\star)(t_n)$ and $\dot{\mathbf{v}}(t_n)$ is its prediction.

Though $\boldsymbol{\sigma}_2$ can be estimated from (4.46), it represents the noise associated with the latent state $\mathbf{v}$, which does not physically exist and therefore lacks a true value. It can also be assumed to be a diagonal matrix in the form $c\mathbf{I}_{d_v}$, where $c\in\mathbb{R}$ is a scalar and $\mathbf{I}_{d_v}\in\mathbb{R}^{d_v\times d_v}$ is the identity matrix. In addition, $\boldsymbol{\sigma}_2$ in the analytic formulae of (4.39) acts as a bias term, and thus c can be treated as either a trainable parameter or a constant. In this work, the scalar c is manually set as a constant.

4.2.5.2 Uncertianty Quantification for Data Assimilation

It is worth noting that the uncertainty quantification of the original unobserved state variables $\mathbf{u}_2$ poses a challenge, which essentially requires mapping the conditional Gaussian distribution of the latent state variables $\mathbf{v}$ through the nonlinear decoder $\boldsymbol{\psi}$. For a weakly nonlinear decoder, this could be achieved efficiently via accessing the Jacobian of the decoder. In cases where the decoder is strongly nonlinear but the dimension of the latent space is relatively low, the unscented transform can serve as an effective method to estimate the uncertainties in $\mathbf{u}_2$. However, the learned decoder of the CGKN model may exhibit strong nonlinearity and operate in a high-dimensional latent space, which prevents an efficient quantification of the uncertainties in $\mathbf{u}_2$ based on existing methods.

To efficiently quantify the uncertainties associated with the posterior mean of the unobserved state $\mathbf{u}_2$, this work explores a post-processing tool based on residual analysis. Residual analysis [359–361] is a method used in statistical modeling and stochastic computing to assess how well a model fits the observed data. With a trained CGKN model, we can perform efficient data assimilation via (4.35) and the decoder $\boldsymbol{\psi}$ to obtain posterior mean $\{\boldsymbol{\mu}(t_n)\}_{n=0}^N$ of unobserved state $\mathbf{u}_2$. The residual $\mathbf{r}$ is then calculated as the absolute difference between true data $\mathbf{u}_2^\star$ and the DA posterior mean $\boldsymbol{\mu}$, i.e., $\mathbf{r}(t_n) = |\mathbf{u}_2^\star(t_n) - \boldsymbol{\mu}(t_n)|$, for $n = 0, 1, ..., N$. The residual $\mathbf{r}$ is used to indicate the uncertainties associated with the posterior mean $\boldsymbol{\mu}$. In practice, the residual $\mathbf{r}$ is assumed to be a function of observed state variables $\mathbf{u}_1$, and the function can be approximated by an auxiliary neural network, with the standard regression task based on a dataset of $\{(\mathbf{u}_1^\star(t_n), \mathbf{r}(t_n))\}_{n=0}^N$ that are available from a trained CGKN model. Under Gaussian assumption, the output of the trained auxiliary neural network can be regarded as estimated standard deviation of the associated posterior mean from maximum likelihood estimation (MLE).

4.2.6 Numerical Experiments

To demonstrate the capability of CGKN in both state prediction and DA, we test it on several complex dynamical systems and compare its performance against other models. Specifically, the tested systems include the projected stochastic Burgers–Sivashinsky equation, which exhibits strong intermittency and extreme events, the Lorenz 96 system with 40 state variables, and a multiscale non-Gaussian climate phenomenon — the El Niño Southern Oscillation (ENSO).

The models that have been tested for each system are summarized as follows:

(i) The true model. This corresponds to the governing equation of the complex dynamical systems in (4.29). State predictions are made by averaging ensemble simulations from any given initial state, while DA results are obtained using the ensemble Kalman-Bucy filter (EnKBF) [72] based on data from observed states. In this work, the EnKBF is employed exclusively by this model for DA, while other models use the analytic DA formulae from (4.39) to ensure a fair comparison. The test results from the true model serve as a benchmark for optimal performance, though they may not always be the best in practical applications.

(ii) Conditional Gaussian Koopman network (CGKN). This is the model depicted in Figure 4.13(a). Without requiring any prior knowledge of the governing equations of the true complex dynamical systems, the CGKN model demonstrates performance comparable to the true model in both state prediction and DA. Furthermore, it significantly improves the efficiency of DA due to the analytic DA formulae provided in (4.39).

(iii) Standard Koopman network (Standard KoopNet). This model applies the standard Koopman theory to the dynamics of the unobserved state $\mathbf{u}_2$, resulting in a linear governing equation for the latent state variables $\mathbf{v}$, rather than the nonlinear but conditionally linear structure of $\mathbf{v}$ found in the CGKN model. It is worth highlighting that this model still retains a strongly nonlinear structure in the dynamics of $\mathbf{u}_1$. The detailed form of the standard Koopman network is:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{u}_1}{\mathrm{d}t} &= \mathbf{f}_1(\mathbf{u}_1) + \mathbf{g}_1(\mathbf{u}_1)\mathbf{v} + \boldsymbol{\sigma}_1\dot{\mathbf{W}}_1, \\ \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} &= \mathbf{F}_2 + \mathbf{G}_2\mathbf{v} + \boldsymbol{\sigma}_2\dot{\mathbf{W}}_2, \end{aligned} \tag{4.47}$$

where $\mathbf{F}_2 \in \mathbb{R}^{\mathrm{d_v}}$ and $\mathbf{G}_2 \in \mathbb{R}^{\mathrm{d_v}\times\mathrm{d_v}}$ are two parameter matrices with other components staying the same as the CGKN model in (4.35). This standard application of the Koopman operator remains to impose a conditional Gaussian structure on the model, but the linear governing equation for $\mathbf{v}$ overlooks the contribution of the observed state $\mathbf{u}_1$. Numerical results show that, in comparison to the standard KoopNet model, the CGKN model's conditional linear structure in the governing equation of $\mathbf{v}$ can further enhance its expressiveness. The comparison between the CGKN and the

standard KoopNet model highlights the crucial role of the conditional linear, yet still nonlinear, structure in the governing equation of $\mathbf{v}$ in the former. The standard KoopNet can be viewed as a simplified version of CGKN where the latent dynamics do not explicitly depend on the observed states. If the true dynamical system possesses this characteristic, then the expressiveness of the standard KoopNet is sufficient for it to perform comparably to CGKN.

(iv) Conditional Gaussian regression (CG-Reg). This is a regression model from system identification that involves two steps: (i) identifying significant basis functions from a library of system dynamics using a causal-based metric called causation entropy [201–204, 221, 362], and (ii) estimating the coefficients of these significant basis functions via the least squares method or maximum likelihood estimation. The library is specifically designed to ensure that the resulting regression model conforms to the structure of the conditional Gaussian nonlinear system in (4.35), allowing the analytic formulae in (4.39) to still be applied for DA [20]. In this work, this specially designed library is referred to as the CG-library, which requires that the unobserved state variables $\mathbf{u}_2$ appear linearly in all the basis functions. The detailed form of the CG-Reg model is as follows:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{u}_1}{\mathrm{d}t} &= \boldsymbol{\Xi}_1\boldsymbol{\Phi} + \boldsymbol{\sigma}_1\dot{\mathbf{W}}_1, \\ \frac{\mathrm{d}\mathbf{u}_2}{\mathrm{d}t} &= \boldsymbol{\Xi}_2\boldsymbol{\Phi} + \boldsymbol{\sigma}_2\dot{\mathbf{W}}_2, \end{aligned} \tag{4.48}$$

where $\boldsymbol{\Phi} = [\phi_1(\mathbf{u}_1, \mathbf{u}_2), \ldots, \phi_M(\mathbf{u}_1, \mathbf{u}_2)]^T \in \mathbb{R}^M$ is a CG-library with $\mathbf{u}_2$ incorporated linearly in each basis function ϕ_i while $\boldsymbol{\Xi}_1 \in \mathbb{R}^{d_{\mathbf{u}_1} \times M}$ and $\boldsymbol{\Xi}_2 \in \mathbb{R}^{d_{\mathbf{u}_2} \times M}$ are two matrices storing the coefficients of the basis functions in the CG-Library $\boldsymbol{\Phi}$. Although the CG-Reg model may have better interpretability, it often exhibits worse performance of state prediction and DA compared to the CGKN model, mainly due to the fact that some important basis functions may be missing in the CG-library for real-world applications. This comparison highlights the expressiveness of the neural networks in the CGKN model.

(v) Deep neural network (DNN): It is a black-box surrogate model based on neural

networks. It reads:

$$\frac{d\mathbf{u}}{dt} = \mathrm{NN}(\mathbf{u}) + \boldsymbol{\sigma}\dot{\mathbf{W}}, \tag{4.49}$$

where $\mathbf{u}$ is the original system state variables including both the observed and unobserved ones. This DNN model does not follow the conditional Gaussian structure, and therefore, the analytic formulae in (4.39) cannot be applied for efficient DA. Numerical results demonstrate that the DNN model achieves state forecast performance comparable to that of the CGKN model; however, the additional key advantage of the CGKN model lies in its unique efficiency in DA.

For all numerical test cases, we assume that both the training and testing data include observed and unobserved state variables, although the inference of a trained CGKN does not require unobserved state variables. To evaluate the performance of models in both state prediction and DA across all examples, the normalized root mean squared error (NRMSE) is used:

$$\mathrm{NRMSE} := \frac{1}{M}\sum_{i=1}^{M} \frac{\sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{x}_i^\star(t_n) - \mathbf{x}_i(t_n)\right)^2}}{\mathrm{std}(\mathbf{x}_i^\star)}, \tag{4.50}$$

where $\mathbf{x}^\star(t_n) \in \mathbb{R}^M$ is the true variables at the time step t_n, $\mathbf{x}(t_n)$ is the approximated variables, $\mathbf{x}_i(t_n)$ denotes the i-th variable, and $\mathrm{std}(\cdot)$ corresponds to the standard deviation estimated by the samples from different time steps $t_n = 1, ..., N$. The NRMSE between true state variables and predictive ones is referred to as the forecast error, and the NRMSE between the true state variables and estimated state variables from DA is referred to as the DA error. Both forecast error and DA error are calculated using test data. The forecast error is calculated based on the same total forecast time steps in training. In contrast, the DA error is calculated over the whole time range of the test dataset. The test results for all models and all numerical examples are summarized in Table 4.8.

4.2.6.1 The Projected Stochastic Burgers–Sivashinsky Equation: A Strongly Nonlinear System with Intermittency and Extreme Events

The Fourier-Galerkin projection of the stochastic Burgers–Sivashinsky equation

$$\partial_t u = \left(\nu\partial_{xx}u + \lambda u - u\partial_x u\right) + \dot{W}(t,x)$$

Table 4.8: Test results of state forecast and data assimilation of all models for each example. The errors are the normalized root mean squared error (NRMSE) between true values and approximated values.

Models \ Examples	PSBSE		Lorenz 96		ENSO PDE	
	Forecast Error	DA Error	Forecast Error	DA Error	Forecast Error	DA Error
True Model	2.4566e-01	6.9466e-01	4.3874e-02	6.9582e-02	6.5406e-01	—
CGKN	2.8389e-01	7.2776e-01	6.3738e-02	9.4883e-02	6.4762e-01	1.3518e-01
Standard KoopNet	3.7600e-01	7.5069e-01	3.0589e-01	2.7252e-01	6.8473e-01	2.3490e-01
CG-Reg	2.7766e-01	1.2884e+00	2.7631e-01	3.5670e-01	9.8751e-01	NaN
DNN	2.4728e-01	—	4.5174e-02	—	6.4845e-01	—

subject to homogeneous Dirichlet boundary conditions is the so-called projected stochastic Burgers–Sivashinsky equation (PSBSE) [333]:

$$\begin{aligned} \frac{dx}{dt} &= \beta_x x + \alpha x y + \alpha y z + \sigma_x \dot{W}_x, \\ \frac{dy}{dt} &= \beta_y y - \alpha x^2 + 2\alpha x z + \sigma_y \dot{W}_y, \\ \frac{dz}{dt} &= \beta_z z - 3\alpha x y + \sigma_z \dot{W}_z. \end{aligned} \tag{4.51}$$

The PSBSE in (4.51) is a stochastic differential equation with energy-conserving quadratic terms and additive Gaussian white noise. The coefficient of the linear terms β_x is positive to introduce instability into the system, and β_y and β_z are negative to linearly damp the system. The coefficient $\alpha > 0$ controls the non-linearity of the system. In this example, the simulation parameters are set as:

$$\begin{gathered} \beta_x = 0.2, \quad \beta_y = -0.3, \quad \beta_z = -0.5, \quad \alpha = 5, \\ \sigma_x = 0.3, \quad \sigma_y = 0.5, \quad \sigma_z = 0.5, \\ \Delta t = 0.001, \quad \text{and} \quad [x_0, y_0, z_0]^\mathsf{T} = [1, 1, 1]^\mathsf{T}. \end{gathered} \tag{4.52}$$

The simulation results of 1000 time units are generated and provide the data with a sub-sampling step size $\Delta t = 0.01$ time unit, where the first 800 units are utilized for training and the remaining 200 units are employed for testing. Figure 4.14 shows part of the simulation of PSBSE in panel (a), and the probability density functions (PDFs) and auto-correlation functions (ACFs) in panels (b) and (c), respectively. The selection of parameters plays a crucial role in the system's behavior: the coefficient of linear terms $\beta_x = 0.2$ introduces instability, while $\beta_y = -0.3$ and $\beta_y = -0.5$ linearly dampens the system. Additionally, setting $\alpha = 5$

contributes significant non-linearity to the system. Under the parameter settings, the true system demonstrates several complex characteristics, including intermittency, extreme events, and non-Gaussian statistics, as illustrated in Figure 3.1. Among the three states, x is set as the observed state, while y and z are set as unobserved states in this test case. This setup of observed and unobserved states aligns with real-world applications since the state x is the largest scale variable in the system.

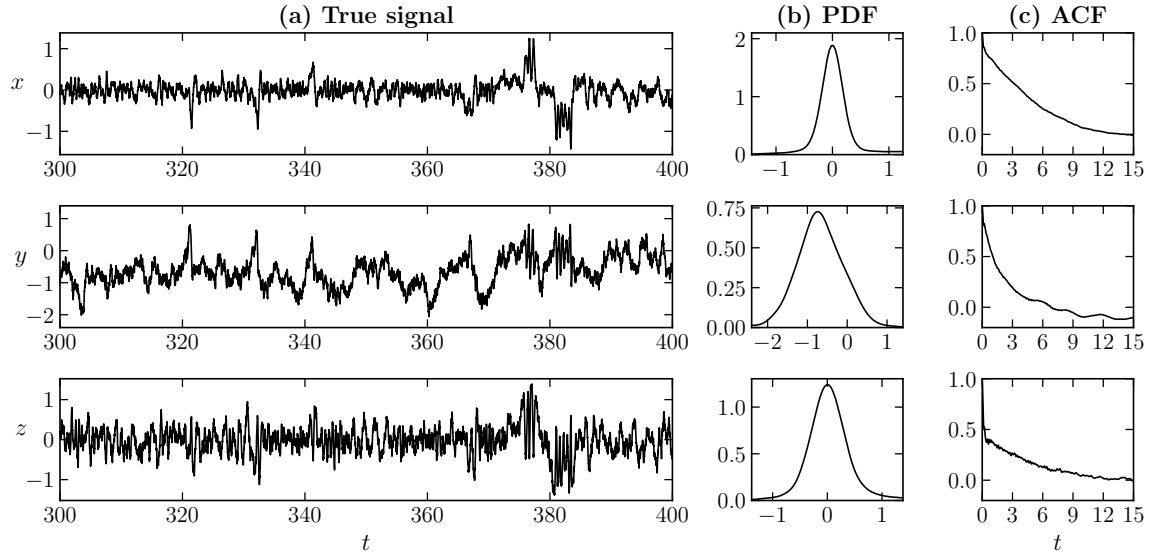

Figure 4.14: Simulation results of the projected stochastic Burgers–Sivashinsky equation, with (a) time series of each state variable, (b) the PDFs of the corresponding state variable, and (c) the ACFs of the corresponding state variable. It should be noted that the PDFs and ACFs are estimated from much longer simulations than the one presented in panel (a).

With the encoder $\boldsymbol{\varphi}$, the decoder $\boldsymbol{\psi}$ and the neural network $\boldsymbol{\eta}(x)$ that outputs $f_1(x)$, $\mathbf{g}_1(x)$, $\mathbf{f}_2(x)$, and $\mathbf{g}_2(x)$, the CGKN model for this example is written as:

$$\begin{aligned} \frac{dx}{dt} &= f_1(x) + \mathbf{g}_1(x)\mathbf{v} + \sigma_x \dot{W}_x, \\ \frac{d\mathbf{v}}{dt} &= \mathbf{f}_2(x) + \mathbf{g}_2(x)\mathbf{v} + \boldsymbol{\sigma}_{\mathbf{v}} \dot{\mathbf{W}}_{\mathbf{v}}, \end{aligned} \tag{4.53}$$

where the latent state variables $\mathbf{v}$ are transformed from unobserved state variables $[y, z]^T$ via the encoder $\boldsymbol{\varphi}$. The dimension of $\mathbf{v}$ is set as 10 in this test case, and the sub-networks in the CGKN model are $\boldsymbol{\varphi} : \mathbb{R}^2 \mapsto \mathbb{R}^{10}$, $\boldsymbol{\psi} : \mathbb{R}^{10} \mapsto \mathbb{R}^2$, and $\boldsymbol{\eta} : \mathbb{R}^1 \mapsto \mathbb{R}^{121}$ with number of parameters

$\boldsymbol{\theta}_{\boldsymbol{\varphi}}$, $\boldsymbol{\theta}_{\boldsymbol{\psi}}$, and $\boldsymbol{\theta}_{\boldsymbol{\eta}}$ be 8778, 8770, and 5785 respectively. The fully connected neural networks are used in CGKN, with the encoder $\boldsymbol{\varphi}$, decoder $\boldsymbol{\psi}$ and neural networks $\boldsymbol{\eta}$ each consisting of five layers. The ReLU activation function is applied to all networks in CGKN. The training settings of the CGKN include: state prediction time range $[t_0, t_{N_s}]$ is 0.1 time units, and the DA time range $[t_0, t_{N_l}]$ is 50 time units with the warm-up period set as $t_{N_b} = 3$ time units. The weights in the target loss function are set as $\lambda_{AE} = 1/d_{\mathbf{u}_2}$, $\lambda_{\mathbf{u}} = \lambda_{\mathbf{v}} = 1/d_{\mathbf{u}}$, and $\lambda_{DA} = 1/d_{\mathbf{u}_2}$. The gradient descent optimizer is set as Adam with an initial learning rate of 1e-3 and a cosine annealing scheduler.

We investigate the performance of both state prediction and DA of all models on the test data and report the results in Table 4.8. The forecast error is calculated by 0.1 time units as the forecast time range and the DA Error is based on 200 time units as the DA time range. For the true model, the ensemble number is set to 100 for both the ensemble mean forecast and the EnKBF. The error in the true model primarily arises from intrinsic random noise in the system, along with sampling errors of the ensemble methods. Consequently, the test results of the true model serve as a benchmark for evaluating the performance of other models. Among all other models, the CGKN model provides the best overall performance in both state forecast and DA. Although the DNN model slightly outperforms the CGKN model for state forecast, its black-box nonlinear architecture demands more computationally expensive DA techniques, while the CGKN model can exploit the analytic formulae of (4.39) for efficient DA. The forecast error from the CG-Reg model is slightly better than that of the CGKN model. The main reason is that, except for the nonlinear yz term, all other terms in the true system of (4.51) can still be successfully identified by the CG-Reg model if all those terms have been included in the CG library. On the other hand, the CG-Reg model provides the worst performance of DA, indicating that the nonlinear term of yz in the governing equation of x plays an important role in the probabilistic relation between the observed state variable x and the unobserved ones y and z. Regarding the standard KoopNet model, the state forecast error is expected to be larger than that of the CGKN model, primarily due to the omission of the contribution from the observed state variable x in the governing equations of the unobserved state variables.

Figure 4.15 presents the DA results from the four models for estimating the unobserved states $[y, z]^\top$ based on the observed state x. To obtain the posterior mean estimates, the EnKBF is employed for the true model, while the analytic formulae in (4.39) are used for the other three models. The uncertainty regions for the true model and the CG-Reg model are derived

from the EnKBF and the covariance evolution formula in (4.39), respectively. In contrast, the uncertainty regions for the CGKN and the standard Koopman network are obtained using the residual analysis method described in Section 4.2.3.2. The comparable performance of the posterior mean estimation to that of the true model underscores the effectiveness of the CGKN in DA. Furthermore, the estimated uncertainty region encompasses most of the results from the true model, even though it remains narrower than that produced by the true model with EnKBF. In this numerical test case, the DA results from the standard KoopNet model shown in Figure 4.15(c) are comparable to those of the CGKN model, while the results from the CG-Reg model in Figure 4.15(d) are noticeably inferior to those of the other models, particularly for the unobserved state variable z.

4.2.6.2 Lorenz 96 System: A Forty-Dimensional Chaotic System

As a widely used numerical example for DA and climate modeling, the governing equation of the single-scale Lorenz 96 system with stochastic noise is as follows [253, 254, 363]:

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F + \sigma \dot{W}_i, \quad i = 1, 2, 3, \ldots, I, \tag{4.54}$$

where $I = 40$, $F = 8$ and $\sigma = 0.5$. Under the specified parameter settings, the Lorenz 96 system exhibits chaotic and wave-like behaviors, allowing it to be regarded as a discretization of atmospheric flow on a latitude circle. Therefore, the current choice of parameters makes the system particularly useful for studying weather patterns and atmospheric dynamics in a controlled yet realistic setting. We simulate the system for 500 time units with the time step size $\Delta t = 10^{-3}$ with the above parameters. With a sub-sampling time step size $\Delta t = 0.01$, the first 300 time units are used as training data, and the rest 200 time units are used for testing. Part of the spatial-temporal simulation results of the true system, including the time series, the PDF, and the ACF of the state variable x_2, are shown in Figure 4.16. Since the state variables in this Lorenz 96 system are statistically homogeneous, i.e., each state variable possesses identical statistical properties, the results of other state variables closely resemble the ones of x_2 shown in Figure 4.16. The PDF and ACF are calculated based on all the 300 time units of training data. In addition to the statistical homogeneity, locality accounts for another feature of the Lorenz 96 system, i.e., the dynamics of a state variable only depend on its nearby state variables. In this numerical test case, a classic DA setup is employed where every second state of the Lorenz 96 system was designated unobserved. More specifically,

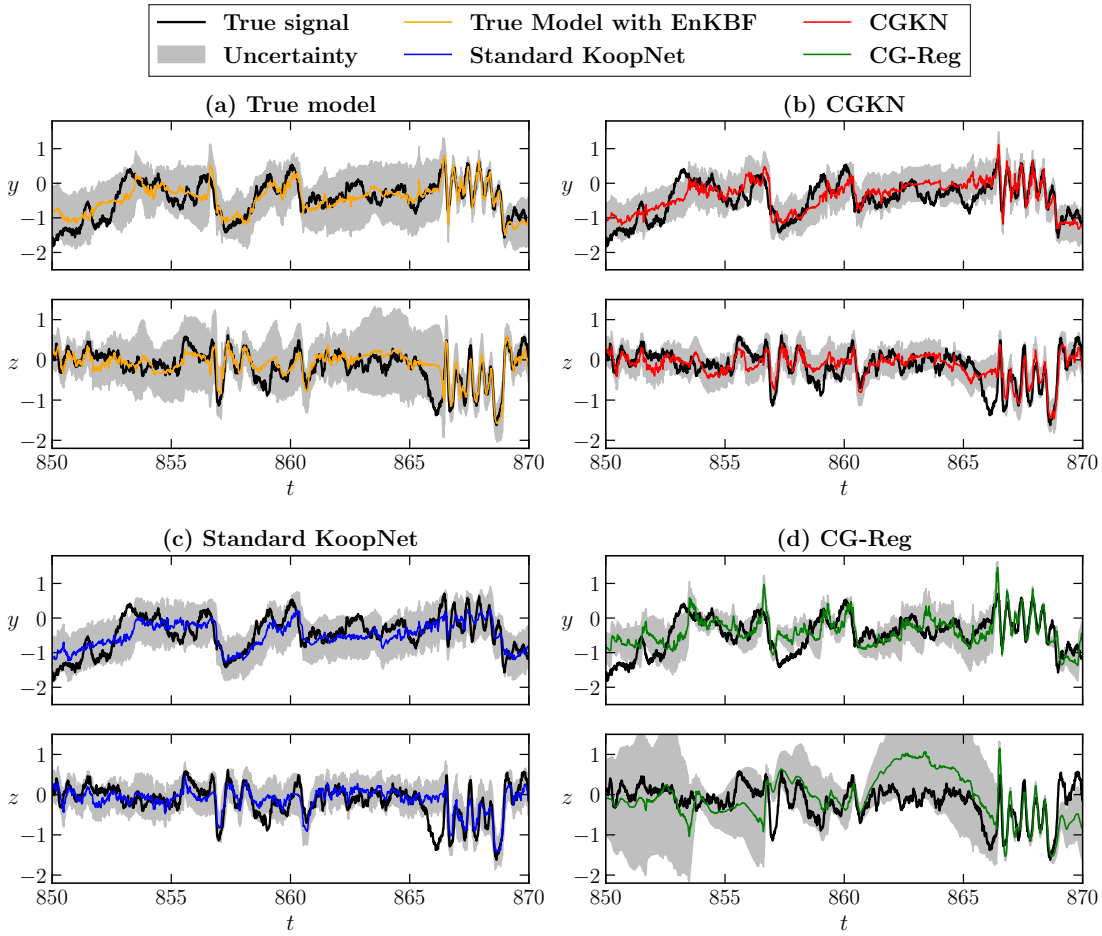

Figure 4.15: DA results of the projected stochastic Burgers–Sivashinsky equation. EnKBF is used for the true model and analytic formulae are used for the other three models. The uncertainties are indicated by the grey-colored regions, which correspond to two estimated standard deviations from the posterior mean. The uncertainty area of the CGKN model and the standard KoopNet model are estimated based on the method of residual analysis.

the system is partially observed with the observed state variables being odd indexed, i.e., $\mathbf{u}_1 = [x_1, x_3, \ldots, x_{39}]^T$, and the unobserved ones being even indexed, i.e., $\mathbf{u}_2 = [x_2, x_4, \ldots, x_{40}]^T$. From this setup, we demonstrate that the physics information, including homogeneous property and local spatial structure, can be compiled into CGKN, which forces the model to be consistent with prior conditions and reduces computational costs.

In this work, locality and homogeneity are both incorporated into the models if possible. More specifically, we assume that dx_i/dt only depends on the local state variable x_i and two nearby state variables $[x_{i-2}, x_{i-1}, x_{i+1}, x_{i+2}]^T$. On the other hand, the homogeneity indicates that the model that approximates the dynamics dx_i/dt based on its dependent state variables is consistent for every i. It is worth noting that the models with conditional Gaussian structure only allow incorporating the homogeneity of the model form for the observed state variables and the unobserved ones separately, while the training data can still inform the models that all state variables of the true system are statistically homogeneous.

In the CGKN model, the autoencoder is also designed to account for locality and homogeneity. For instance, the unobserved state variables x_{2i-2} and x_{2i+1} nearby the odd-indexed observed state variable x_{2i-1} are transformed to latent state $\mathbf{v}^{(i)} = [\mathbf{v}_1^{(i)}, \mathbf{v}_2^{(i)}, \ldots, \mathbf{v}_J^{(i)}]$ via encoder $\boldsymbol{\varphi}$ for $i = 1, 2, \ldots, 20$.

With the encoder $\boldsymbol{\varphi}$, decoder $\boldsymbol{\psi}$ and the DNNs $\boldsymbol{\eta}$ including $\mathrm{NN}^{(\mathbf{v})} = [\mathrm{NN}^{(\mathbf{v}_1)}, \mathrm{NN}^{(\mathbf{v}_2)}, \ldots, \mathrm{NN}^{(\mathbf{v}_J)}]$ and $\mathrm{NN}^{(\mathbf{u}_1)}$, the framework of CGKN with locality and homogeneity reads:

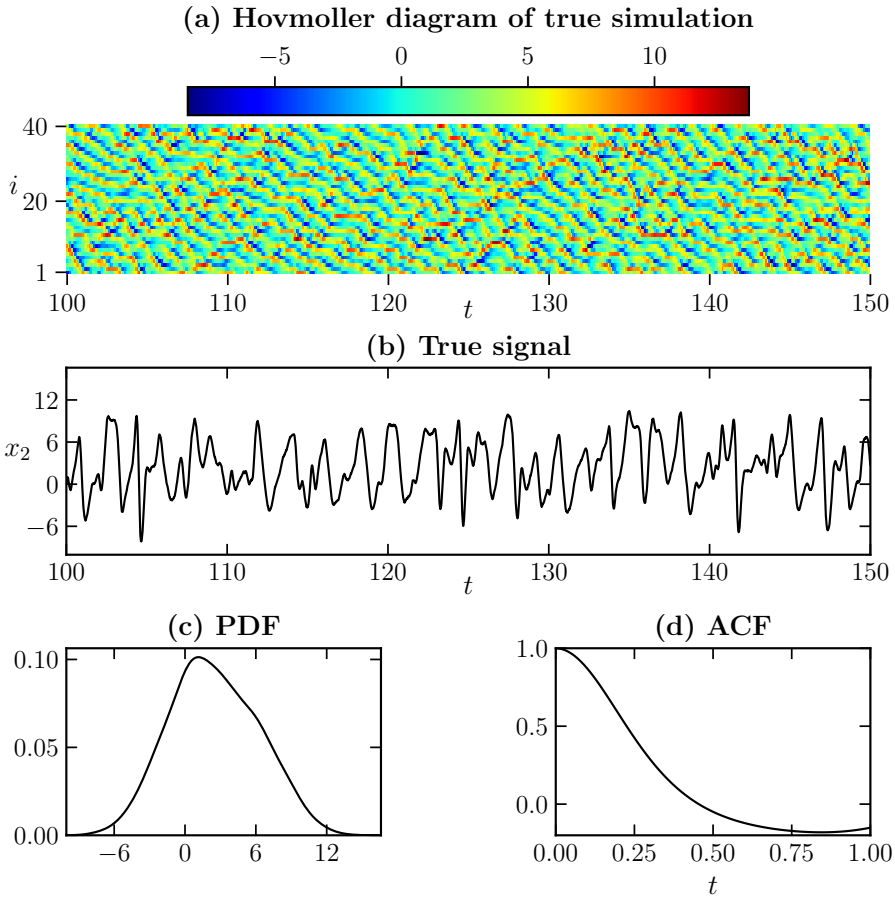

Figure 4.16: Simulation results and statistics of the Lorenz 96 system, with (a) the Hovmoller diagram of the spatiotemporal patterns, (b) time series of x_2, (c) the PDF of x_2, and (d) the ACF of x_2. Note that the PDF and ACF are estimated from a simulation longer than the one presented in panel (b). The behavior of other state variables is similar to x_2 since the system is statistically homogeneous.

$$
\begin{aligned}
\frac{dx_{2i-1}}{dt} =& NN_0^{(\mathbf{u}_1)} + \\
& NN_1^{(\mathbf{u}_1)}\mathbf{v}_1^{(i-1)} + NN_2^{(\mathbf{u}_1)}\mathbf{v}_2^{(i-1)} + \cdots + NN_J^{(\mathbf{u}_1)}\mathbf{v}_J^{(i-1)} + \\
& NN_{J+1}^{(\mathbf{u}_1)}\mathbf{v}_1^{(i)} + NN_{J+2}^{(\mathbf{u}_1)}\mathbf{v}_2^{(i)} + \cdots + NN_{2J}^{(\mathbf{u}_1)}\mathbf{v}_J^{(i)} + \\
& NN_{2J+1}^{(\mathbf{u}_1)}\mathbf{v}_1^{(i+1)} + NN_{2J+2}^{(\mathbf{u}_1)}\mathbf{v}_2^{(i+1)} + \cdots + NN_{3J}^{(\mathbf{u}_1)}\mathbf{v}_J^{(i+1)} + \\
& \sigma_x \dot{W}_x \\
\frac{d\mathbf{v}_j^{(i)}}{dt} =& NN_0^{(\mathbf{v}_j)} + \\
& NN_1^{(\mathbf{v}_j)}\mathbf{v}_1^{(i-2)} + NN_2^{(\mathbf{v}_j)}\mathbf{v}_2^{(i-2)} + \cdots + NN_J^{(\mathbf{v}_j)}\mathbf{v}_J^{(i-2)} + \\
& NN_{J+1}^{(\mathbf{v}_j)}\mathbf{v}_1^{(i-1)} + NN_{J+2}^{(\mathbf{v}_j)}\mathbf{v}_2^{(i-1)} + \cdots + NN_{2J}^{(\mathbf{v}_j)}\mathbf{v}_J^{(i-1)} + \\
& NN_{2J+1}^{(\mathbf{v}_j)}\mathbf{v}_1^{(i)} + NN_{2J+2}^{(\mathbf{v}_j)}\mathbf{v}_2^{(i)} + \cdots + NN_{3J}^{(\mathbf{v}_j)}\mathbf{v}_J^{(i)} + \\
& NN_{3J+1}^{(\mathbf{v}_j)}\mathbf{v}_1^{(i+1)} + NN_{3J+2}^{(\mathbf{v}_j)}\mathbf{v}_2^{(i+1)} + \cdots + NN_{4J}^{(\mathbf{v}_j)}\mathbf{v}_J^{(i+1)} + \\
& NN_{4J+1}^{(\mathbf{v}_j)}\mathbf{v}_1^{(i+2)} + NN_{4J+2}^{(\mathbf{v}_j)}\mathbf{v}_2^{(i+2)} + \cdots + NN_{5J}^{(\mathbf{v}_j)}\mathbf{v}_J^{(i+2)} + \\
& \sigma_{\mathbf{v}} \dot{W}_{\mathbf{v}} \\
& i = 1, \ldots, 20;\ j = 1, 2, \ldots, J.
\end{aligned}
\tag{4.55}
$$

In this test case, the parameter J is selected as 6, and therefore, the dimension of latent state $d_{\mathbf{v}}$ in (4.35) is $20J = 120$. The $NN_i^{(\mathbf{u}_1)}$ stands for the $(i+1)$-th output of the network $NN^{(\mathbf{u}_1)} \in \mathbb{R}^3 \mapsto \mathbb{R}^{19}$ which takes input $(x_{2i-3}, x_{2i-1}, x_{2i+1})$ for dynamics dx_{2i-1}/dt and similar to $NN^{(\mathbf{v}_j)} \in \mathbb{R}^3 \mapsto \mathbb{R}^{31}$ which takes input $(x_{2i-3}, x_{2i-1}, x_{2i+1})$ for dynamics $d\mathbf{v}_j^{(i)}/dt$. Since all the neural networks $NN^{(\mathbf{v}_j)}$ take the same inputs for each $j = 1, 2, \ldots, J$, they are merged into a single neural network $NN^{(\mathbf{v})} :\in \mathbb{R}^3 \mapsto \mathbb{R}^{186}$ which takes input $(x_{2i-3}, x_{2i-1}, x_{2i+1})$ for dynamics $d\mathbf{v}^{(i)}/dt = [d\mathbf{v}_1^{(i)}/dt, d\mathbf{v}_2^{(i)}/dt, \ldots, d\mathbf{v}_J^{(i)}/dt]$. When iterating $NN^{(\mathbf{u}_1)}$ across all dynamics of observed state, the outputs will formalize the $\mathbf{f}_1 \in \mathbb{R}^{20\times 1}$ and $\mathbf{g}_1 \in \mathbb{R}^{20\times 120}$ in (4.35). Similarly, iterating $NN^{(\mathbf{v})}$ across all dynamics of latent state will formalize the $\mathbf{f}_2 \in \mathbb{R}^{120\times 1}$ and $\mathbf{g}_2 \in \mathbb{R}^{120\times 120}$. One thing should be noted is that, due to the locality strategy, the $\mathbf{f}_2$ and $\mathbf{g}_2$ are sparse matrices with zero elements representing non-interaction with the uncorrelated states and non-zero elements filled by the outputs of $NN^{(\mathbf{u}_1)}$ and $NN^{(\mathbf{v})}$ respectively. The sub-networks in the CGKN with locality and homogeneity of this test case are $\boldsymbol{\varphi} : \mathbb{R}^2 \mapsto \mathbb{R}^6$, $\boldsymbol{\psi} : \mathbb{R}^6 \mapsto \mathbb{R}^2$ and $\boldsymbol{\eta} = [NN^{(\mathbf{u}_1)}, NN^{(\mathbf{v})}]$ with the number of parameters $\boldsymbol{\theta}_{\boldsymbol{\varphi}}$, $\boldsymbol{\theta}_{\boldsymbol{\psi}}$ and $\boldsymbol{\theta}_{\boldsymbol{\eta}}$ be 8646, 8942 and 5757, respectively. In the CGKN, the encoder $\boldsymbol{\varphi}$, decoder

$\boldsymbol{\psi}$ and sub-networks $\boldsymbol{\eta}$ including $\mathrm{NN}^{(\mathbf{v})}$ and $\mathrm{NN}^{(\mathbf{u}_1)}$ are all fully connected networks with 5, 5, 4, and 4 layers, respectively. The activation function is chosen as ReLU. The training setup for CGKN is as follows: the state forecast horizon is $t_{N_s} = 0.2$ time units, the DA horizon is $t_{N_l} = 60$ time units with the first $t_{N_b} = 2$ time units be cut out. The weights in the target loss function are set as $\lambda_{AE} = 1/d_{\mathbf{u}_2}$, $\lambda_{\mathbf{u}} = \lambda_{\mathbf{v}} = 1/d_{\mathbf{u}}$, and $\lambda_{DA} = 1/d_{\mathbf{u}_2}$. The gradient descent optimizer is selected as Adam with an initial learning rate of 1e-3 and a cosine annealing scheduler. It is worth noting that the complexity of the CGKN incorporated with the locality and homogeneity does not increase with the dimensionality of the Lorenz 96 system which makes CGKN scalable to even higher dimensions.

The test results for the different models are reported in Table 4.8. For the true model, we use the mean of ensemble simulations for state prediction and employ the EnKBF for DA, with an ensemble size of 100 for both tasks. Among all the other models, the CGKN is the only one that achieves similar performance in both state forecast and DA compared to the true model. In terms of state forecast, the DNN model performs slightly better than the CGKN model. However, the CGKN's advantage lies in its ability to avoid the computationally expensive DA techniques typically required for nonlinear models, making it significantly more efficient for DA. Regarding models that allow efficient DA, the standard KoopNet model slightly outperforms the CG-Reg model; however, both models underperform compared to the CGKN model.

Figure 4.17 illustrates the true state variables alongside the prediction results from different models with a lead time of 0.2 time units. The Hovmoller diagram (namely, the spatiotemporal field) of the true system and the simulation results from the CGKN model indicate that the CGKN successfully captures the overall chaotic behavior and wave patterns. The time series of state variables x_1 and x_2 from the true system, along with the forecast results from the various models, are also presented in Figure 4.17. The results for other observed and unobserved state variables are similar to those of x_1 and x_2 due to the statistical homogeneity of this test case and are thus omitted. Notably, both the standard KoopNet model and the CG-Reg model exhibit discrepancies compared to the truth, while the CGKN model shows results that are more accurate. Although the standard KoopNet model performs reasonably well in forecasting the observed variable x_1, its performance for the unobserved state variable x_2 is inferior to that of the CGKN model. This discrepancy primarily arises from the neglect of the contribution of observed state variables to the dynamics of unobserved state variables in the standard KoopNet model, which is a crucial factor in the true system

due to the quadratic term in (4.54).

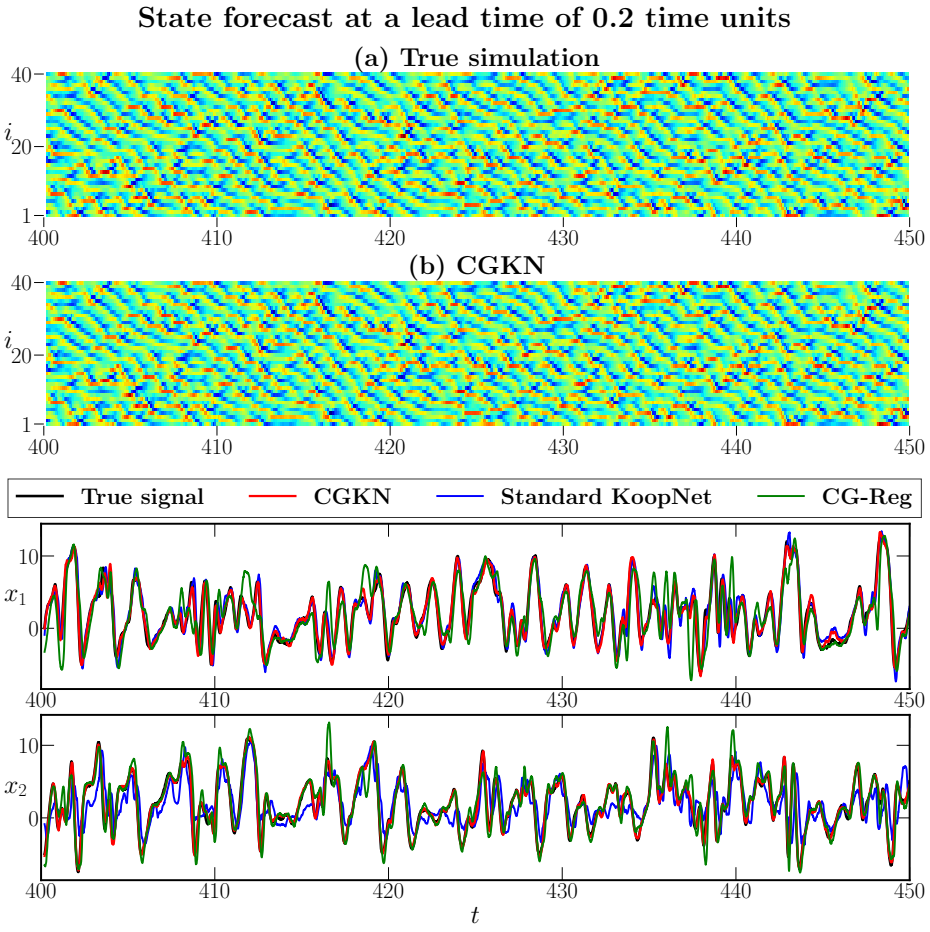

Figure 4.17: State forecast results with a lead time of 0.2 time units. Panel (a) and (b) are the Hovmoller diagram of the true system and results from the CGKN model. The bottom two panels show the true signal of states x_1 and x_2 and the corresponding results from three different models.

Figure 4.18 presents the DA results of the unobserved state x_2. In each panel, the lower sub-figure provides a zoomed-in view of the upper one to clearly display the trajectory and uncertain area. In this case, the DA results from the true model with EnKBF yield a posterior mean that closely aligns with the true signal and a narrow uncertainty area, which is obtained from the estimated covariance of ensemble simulations. The CGKN model shows a good agreement with the true model and outperforms the standard KoopNet model and the CG-Reg model. For the CG-Reg model, the uncertainty area is derived from the

covariance evolution formula with a similar form of (4.39). For both the CGKN model and the standard KoopNet model, the uncertainty area is based on the residual analysis method described in Section 4.2.3.2.

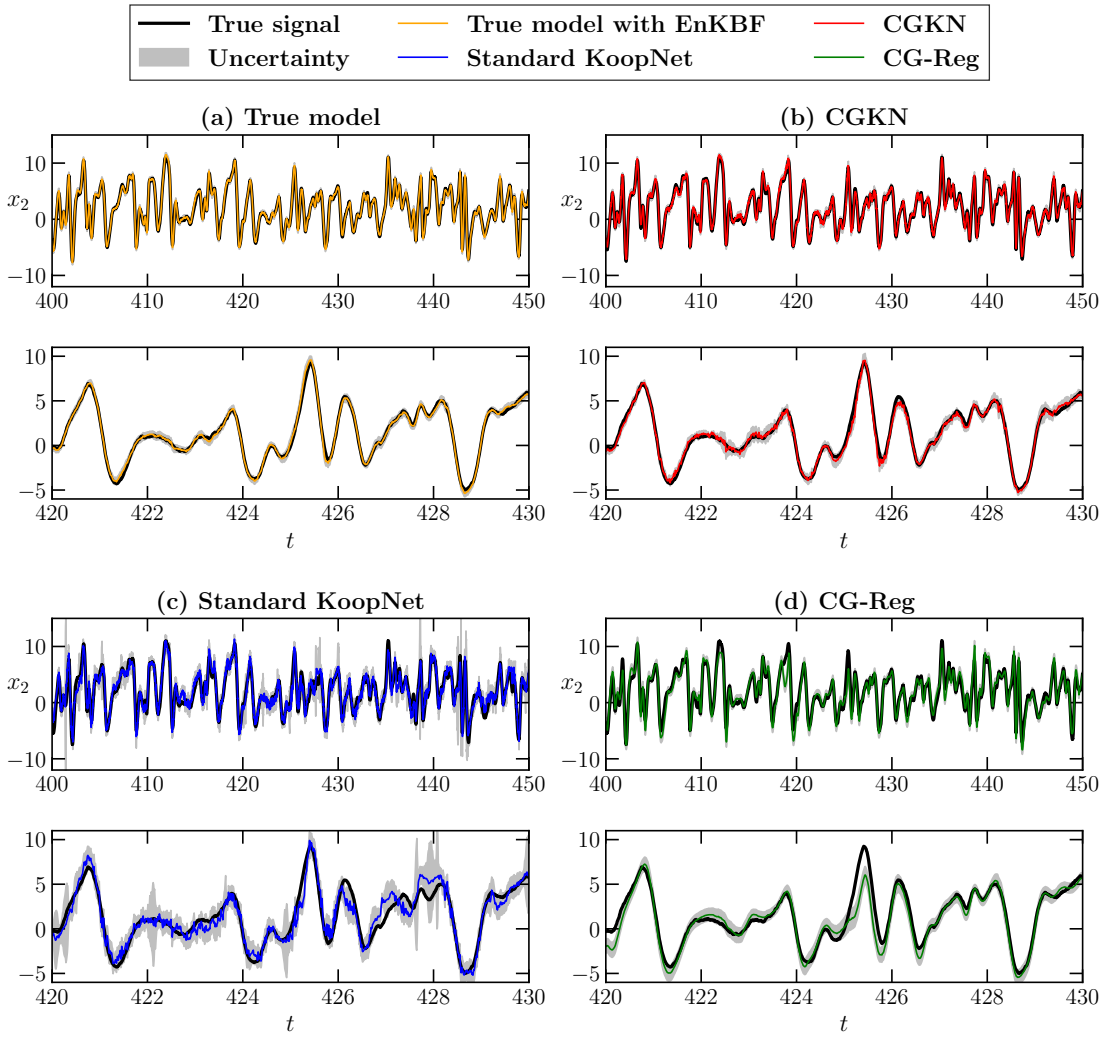

Figure 4.18: DA results of the unobserved state variable x_2 in the Lorenz 96 system. EnKBF is used for the true model, and analytic DA formulae are used for the other three models. The uncertainties are indicated by the grey-colored regions, which correspond to two estimated standard deviations from the posterior mean. The uncertainty area from the CGKN model and the standard KoopNet model are derived from residual analysis.

4.2.6.3 El Niño-Southern Oscillation (ENSO): A Multiscale Non-Gaussian Phenomenon

The El Niño-Southern Oscillation (ENSO) is the most significant source of interannual climate variability in the tropics, characterized by fluctuations in sea surface temperature (SST) in the equatorial Pacific [364–367]. Its warm (El Niño) and cold (La Niña) phases have profound effects on global weather patterns, ecosystems, and socioeconomic systems. Accurate forecasting of ENSO is crucial for improving climate predictions and mitigating the impact of extreme weather events.

A physics-based reference model is employed to generate synthetic ENSO data [368] for training and testing different methods here. This model consists of a set of stochastic partial differential equations (SPDEs) that describe the coupled dynamics of the atmosphere and ocean. It effectively captures many realistic features observed in the real world, including non-Gaussian statistics and variations in spatial patterns, peak intensity, and temporal evolution. Its capability to generate much longer time series than actual observations enables us to train and evaluate the skill of the CGKN on this complex natural phenomenon. More details and the governing equations of the ENSO SPDEs are available in Appendix E.

The physical variables of focus here are SST **T**, thermocline depth **H**, and wind amplitude a_p. The thermocline is a layer in a body of water where temperature decreases rapidly with depth, separating warmer surface waters from colder deep waters. SST and thermocline depth are considered interannual variables, evolving much more slowly than wind, which is classified as intraseasonal variability. SST is a commonly used indicator for measuring ENSO events, and it is strongly coupled with thermocline depth, together forming an irregular oscillator that reflects the interactions between the ocean and the sea surface. Despite its faster timescale, wind acts as an external forcing that drives the ocean and SST variables.

We conducted two simulations, each spanning 2000 years, using the ENSO SPDEs, and spatially averaged the state variables **T** (SST) and state **H** (thermocline depth) across three longitudinal ranges. The resulting discrete state variables are denoted as T_W, T_C, T_E for the SST in the western, central, and eastern Pacific, respectively, along the equator, with the same notation for H_W, H_C and H_E for thermocline depth. It has been demonstrated that these coarse-grained state variables are sufficient to capture the large-scale features of ENSO dynamics [369, 370]. Together with the state variable wind burst amplitude a_p, one 2000-year simulation of all Ts and Hs is used as training data and the other one is for

testing, with both datasets having a time step of $\Delta t = 1/360$ (about 1 day).

Figure 4.19 displays a portion of the training data for T_E, H_W, and a_p in panel (a), along with their corresponding PDFs and ACFs in panels (b) and (c), evaluated using the entire training dataset. In this ENSO test case, the observed state variables are T_W, T_C, T_E, and a_p, while the unobserved ones are H_W, H_C, H_E, with a_p serving as an external variable. The models, including CGKN, are trained on the data of all state variables and aim to forecast both T and H states, while performing DA without access to H data. In practice, SST data is obtained from satellite instruments, while directly measuring thermocline depth is challenging. DA is involved in recovering the thermocline depth in practice. Therefore, estimating thermocline depth from SST in this setup simulates a realistic scenario.

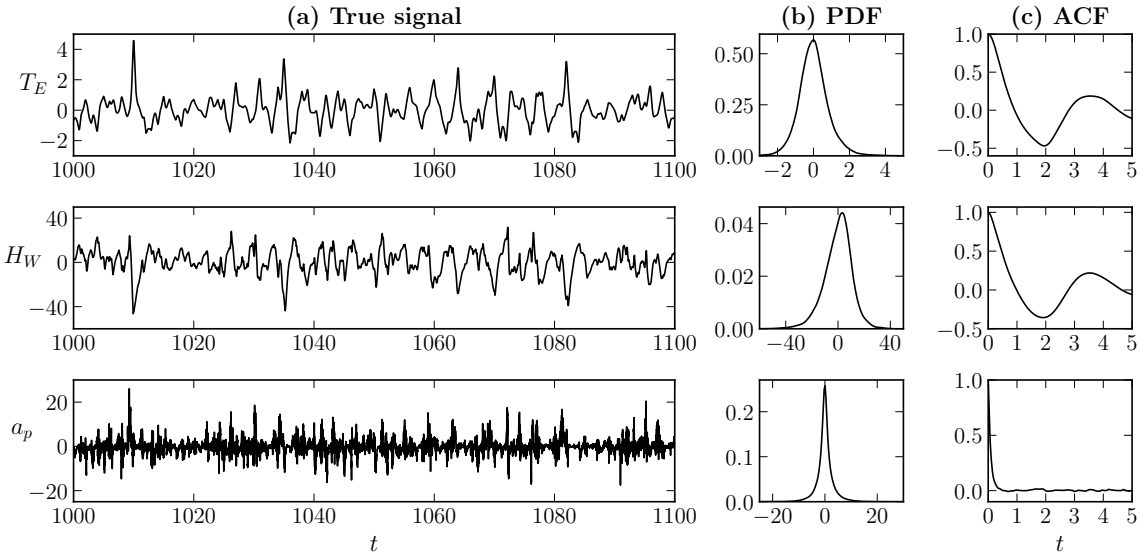

Figure 4.19: Time series and statistics property of states T_E, H_W, a_p derived from the ENSO PDEs. Panel (a): part of time series of each state variable. Panel (b) and (c): probability density function (PDF) and auto-correlation function (ACF) of the corresponding state. It should be noted that the PDFs and ACFs are estimated from a much longer series than the one presented in panel (a).

With the encoder φ, the decoder ψ and a neural network η that outputs $\mathbf{f}_1$, $\mathbf{g}_1$, $\mathbf{f}_2$, and

$\mathbf{g}_2$, the CGKN model for this example is written as:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}[\mathsf{T}_W, \mathsf{T}_C, \mathsf{T}_E, \mathsf{a}_p]^\mathsf{T} &= \mathbf{f}_1(\mathsf{T}_W, \mathsf{T}_C, \mathsf{T}_E, \mathsf{a}_p) + \mathbf{g}_1(\mathsf{T}_W, \mathsf{T}_C, \mathsf{T}_E, \mathsf{a}_p)\mathbf{v} + \boldsymbol{\sigma}_1 \dot{\mathbf{W}}_1, \\ \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} &= \mathbf{f}_2(\mathsf{T}_W, \mathsf{T}_C, \mathsf{T}_E, \mathsf{a}_p) + \mathbf{g}_2(\mathsf{T}_W, \mathsf{T}_C, \mathsf{T}_E, \mathsf{a}_p)\mathbf{v} + \boldsymbol{\sigma}_\mathbf{v} \dot{\mathbf{W}}_\mathbf{v}, \end{aligned} \tag{4.56}$$

where $\mathbf{v} = \boldsymbol{\varphi}(\mathsf{H}_W, \mathsf{H}_C, \mathsf{H}_E)$ and $[\mathsf{H}_W, \mathsf{H}_C, \mathsf{H}_E]^\mathsf{T} = \boldsymbol{\psi}(\mathbf{v})$. The latent state dimension of the autoencoder is set as 10 in this test case. With this choice, the sub-networks in the CGKN model are $\boldsymbol{\varphi} : \mathbb{R}^3 \mapsto \mathbb{R}^{10}$, $\boldsymbol{\psi} : \mathbb{R}^{10} \mapsto \mathbb{R}^3$, $\boldsymbol{\eta} : \mathbb{R}^4 \mapsto \mathbb{R}^{154}$ with number of parameters being 10554, 12970, 12963, respectively. In CGKN, the fully connected neural networks are used for encoder $\boldsymbol{\varphi}$, decoder $\boldsymbol{\psi}$, and sub-network $\boldsymbol{\eta}$, each consisting of six layers with ReLU activation functions. For the training settings, the state forecast time range $[\mathsf{t}_0, \mathsf{t}_{\mathsf{N}_\mathsf{s}}]$ is set as 1 year, and DA time range $[\mathsf{t}_0, \mathsf{t}_{\mathsf{N}_\mathsf{l}}]$ is 40 years with the warm-up period $\mathsf{t}_{\mathsf{N}_\mathsf{b}}$ being 4 years. The weights in the target loss function are set as $\lambda_{\mathrm{AE}} = 1/\mathsf{d}_{\mathbf{u}_2}$, $\lambda_\mathbf{u} = 1/\mathsf{d}_\mathbf{u}$, $\lambda_\mathbf{v} = 0$ and $\lambda_{\mathrm{DA}} = 1/\mathsf{d}_{\mathbf{u}_2}$. The gradient descent optimizer is selected as Adam with an initial learning rate of 1e-3 and a cosine annealing scheduler.

The test results of all models for this case have been reported in Table 4.8. The ensemble number of the true model for ensemble mean state forecast is set to 100. The DA error from EnKBF has been omitted due to the computational resource required by the ENSO PDEs and it is not the focus of this work. For the performance of state forecast, except for the CG-Reg model, the other models achieve similar results with the standard KoopNet model ranked the lowest. For the DA performance, the CG-Reg model becomes unstable and diverges over the horizon of 2000 years from test data. The DA error of the CGKN model is the lowest, outperforming the performance of the standard KoopNet. In this test case, the CGKN model demonstrates optimal performance in both state forecast and DA compared to all other models.

Figure 4.20 shows the lead time prediction of different models in terms of T_E, which is one of the most widely used ENSO indicator. The panels (a) and (b) display the NRMSE and the pattern correlation between the true signal of state T_E and its predictive states at different lead times. The correlation is a standard measurement in climate science and it is therefore included here. The NRMSE is calculated based on the formula of (4.50) and the

correlation is from the formula:

$$\text{Corr} = \frac{\sum_{n=1}^{N} \left(x^\star(t_n) - \bar{x}^\star\right)\left(x(t_n) - \bar{x}\right)}{\sqrt{\sum_{n=1}^{N} \left(x^\star(t_n) - \bar{x}^\star\right)^2} \sqrt{\sum_{n=1}^{N} \left(x(t_n) - \bar{x}\right)^2}} \tag{4.57}$$

with $x^\star \in \mathbb{R}^N$ is the true values of state time series, x is the approximated values, $x(t_n)$ is the state at time t_n. The $\bar{x}^\star$ and $\bar{x}$ are the time average of true values and approximated values. In this example, the persistence model, a simple and commonly used prediction method for ENSO, is considered. It assumes that ENSO conditions remain unchanged over time and use the current values as future predictions. The persistence serves as a baseline model for evaluating other computational models. The panels showing NRMSE and correlation indicate that both the CGKN model and the standard KoopNet model produce stable and accurate state forecast results across various lead times, demonstrating the effectiveness of these deep learning models. In contrast, the CG-Reg model exhibits a rapid decline in forecast performance due to its omission of certain key dynamics in the parametric forms, while the persistence model performs the worst among all the models. Panels (c)-(d) show the time series of the true state and predictive states at a lead time of 3 months, 6 months, 9 months, and 12 months, respectively. For the lead time prediction at 3 months in panel (c), all the models including persistence can almost follow the true signal. However, with the lead time becoming longer, the deviation and shift pattern of predictions from all models have become more pronounced. For the 12-month lead time prediction in panel (f), although the forecast skill of the CGKN model declines, it still performs the best overall. The model maintains a correlation above 0.5 and significantly outperforms the persistence model.

Figure 4.21 shows the DA results from the CGKN model for estimating the unobserved state H_W, H_C and H_E given trajectories of observed state T_W, T_C, T_E and a_p. The governing equations of the true model for ENSO are stochastic partial differential equations (SPDEs), detailed in Appendix B. The SPDEs are computationally intensive and time-consuming for ensemble methods to perform data assimilation effectively. For the CG-Reg model, the DA result is very unstable and diverges quickly, leading to the finite-time blow-up of the posterior mean and covariance. In addition, the standard KoopNet shows a similar DA result to the CGKN. Therefore, Figure 3.8 primarily focuses on the DA results of CGKN compared to the true signal. The DA posterior mean from the CGKN model is obtained by applying the analytic formulae in (4.39), while the uncertainty areas are determined using

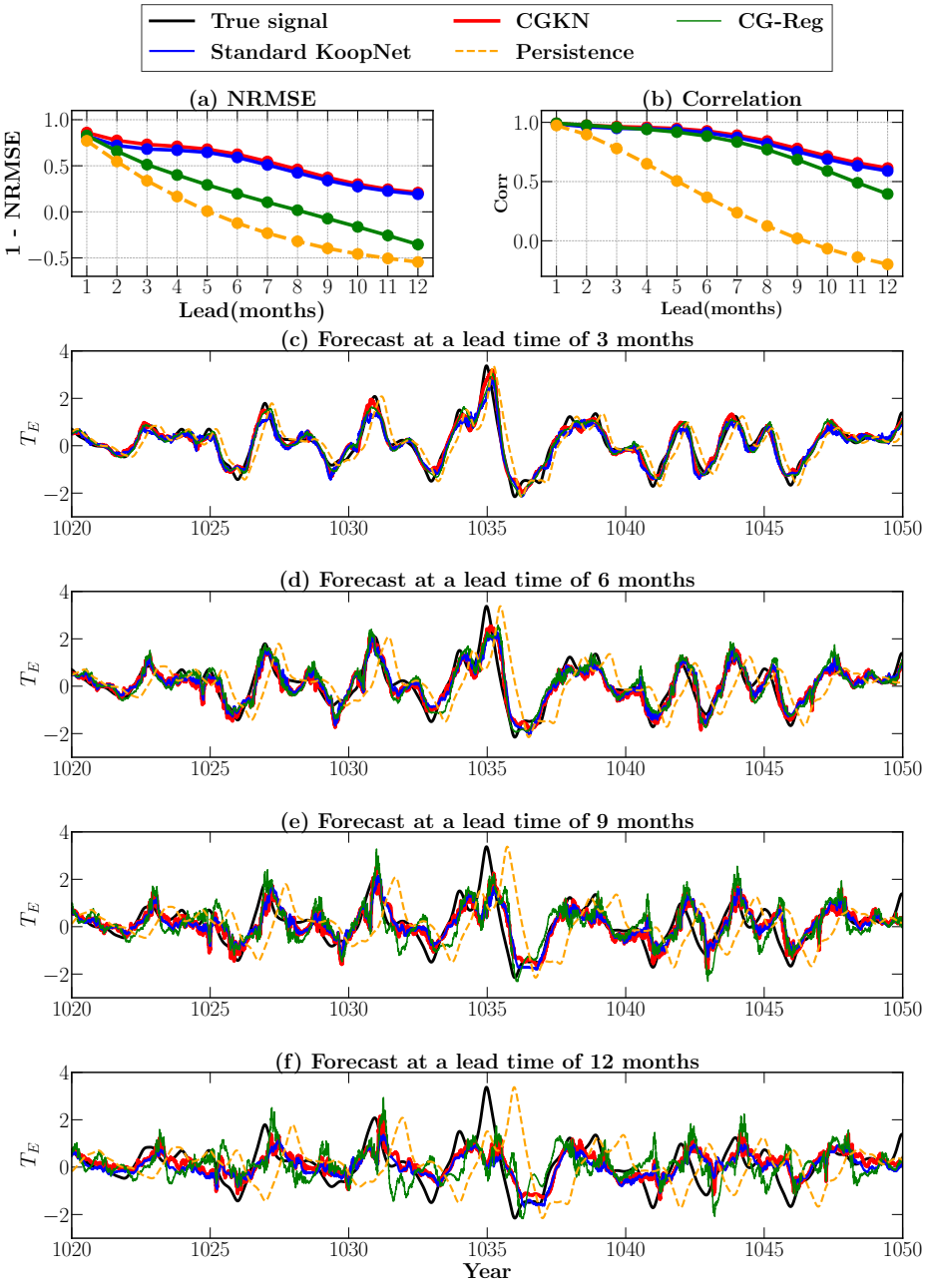

Figure 4.20: Comparison of the lead-time forecast for state T_E from different models. Panel (a) and (b): skill scores including normalized root mean squared error (NRMSE) and correlation of the CGKN model, standard KoopNet model, CG-Reg model, and Persistence. Panel (c) - (f): comparison of the true signal and forecast results from different models at lead times of 3, 6, 9, and 12 months, respectively, over the years 1020 to 1050.

the method of residual analysis described in Section 4.2.3.2. The similarity between the posterior mean from the CGKN model and the true signal confirms its capability in DA. Although the posterior mean from the CGKN model shows does not perfectly match the true signal occasionally, the uncertainty area can still cover the true data for most of those times.

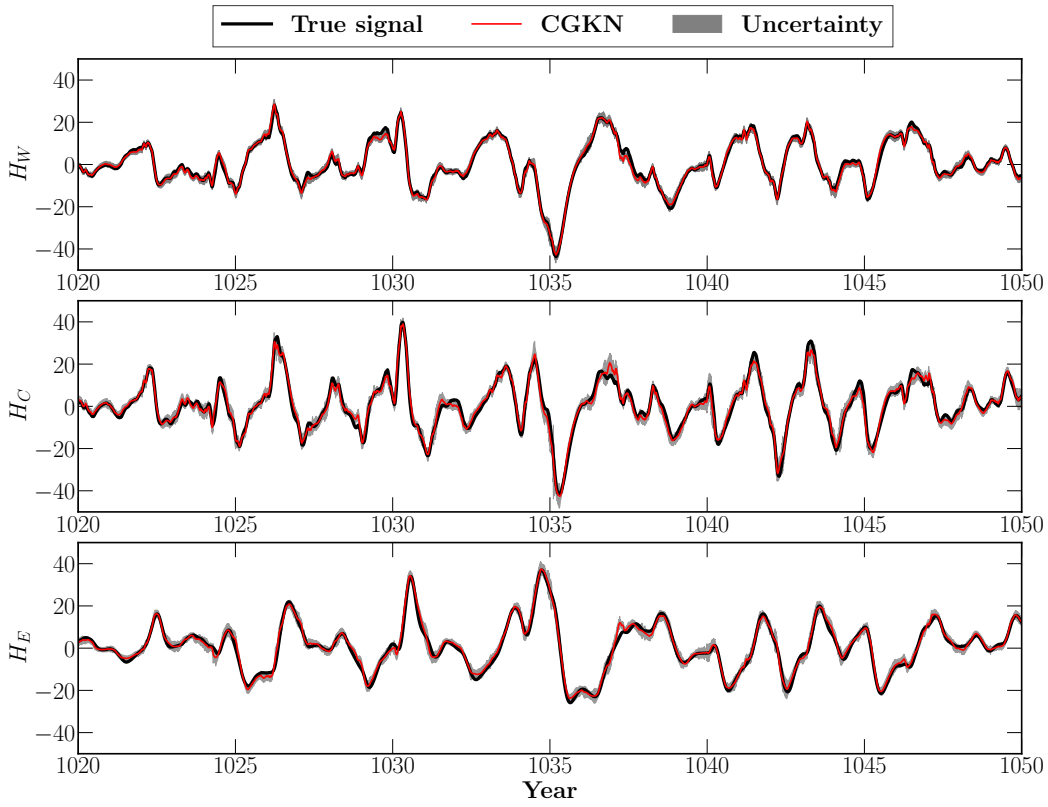

Figure 4.21: DA results of the unobserved state variables H_W, H_C, and H_E derived from physics-based PDEs for ENSO. The DA posterior mean is obtained from applying the analytic DA formulae to the CGKN model and the uncertainty is estimated based on the method of residual analysis. The uncertainties are indicated by the grey-colored regions, which correspond to two estimated standard deviations from the posterior mean.

In addition to comparing the trajectories of the true states and approximated ones from forecast and DA, the spatiotemporal simulation of **T** (SST) and **H** (thermocline) of the true fields and the approximated ones are also presented in Figure 4.22. The approximated fields are obtained from the reconstruction of discrete states Ts and Hs via a bi-variate regression

method:

$$\begin{aligned}\tilde{\mathbf{T}}(x,t) &= r_W(x)T_W(t) + r_C(x)T_C(t) + r_E(x)T_E(t),\\ \tilde{\mathbf{H}}(x,t) &= l_W(x)H_W(t) + l_C(x)H_C(t) + l_E(x)H_E(t),\end{aligned} \tag{4.58}$$

where x is the longitude and t is the temporal variable. The regression coefficients $r_W(x)$, $r_C(x)$, $r_E(x)$, $l_W(x)$, $l_C(x)$, and $l_E(x)$ are determined using the true signals at each grid point of longitude x. The discrete states from either the spatial average of true simulation or inference of the CGKN model are plugged into the regression formula in (4.58) to obtain the reconstructed SST field $\mathbf{T}$ and reconstructed thermocline field $\mathbf{H}$. Figure 4.22 shows the spatial-temporal evolution of true simulations, true reconstruction from spatial-averaged state Ts and Hs, and reconstruction from approximated ones via the CGKN model. In the third column of left panel, the predictive spatial-temporal simulation of $\mathbf{T}$ is reconstructed from 6-month lead time forecast of T_W, T_C and T_E via CGKN. The corresponding time series of T_E is shown in Figure 4.20(d). The comparison to the true PDE simulation confirms the capability of state forecast of the CGKN model since the overall pattern of the evolution of SST can be reproduced. The DA results are shown in the right panel. The spatiotemporal simulation of $\mathbf{H}$ (thermocline) in the third column is reconstructed from the estimated states T_W, T_C, and T_E obtained by applying the analytic DA formulae in (4.39) to the CGKN model. With these accurate DA posterior means of states Hs as shown in Figure 4.21, the successful reconstruction of the spatially continuous $\mathbf{H}$ field is as expected.

4.2.7 Discussion and Conclusion

In this work, we propose a conditional Gaussian Koopman network — CGKN — as a unique deep learning framework for modeling complex dynamical systems that facilitate both accurate forecasts and efficient DA. The proposed framework is inspired by the Koopman theory. It builds upon the conditional Gaussian nonlinear system, transforming any nonlinear system into a modeled system that embeds a conditional linear structure to facilitate efficient analytic formulae of DA. The proper nonlinear transformation is jointly learned with the unknown terms in the transformed conditional Gaussian nonlinear system, based on a total loss function that accounts for the overall performance of the nonlinear transformation, the state prediction, and the DA. Among all the models that support efficient analytic formulae of DA, the numerical examples demonstrate that the CGKN model generally outperforms the standard KoopNet model and the CG-Reg model in both state prediction and DA.

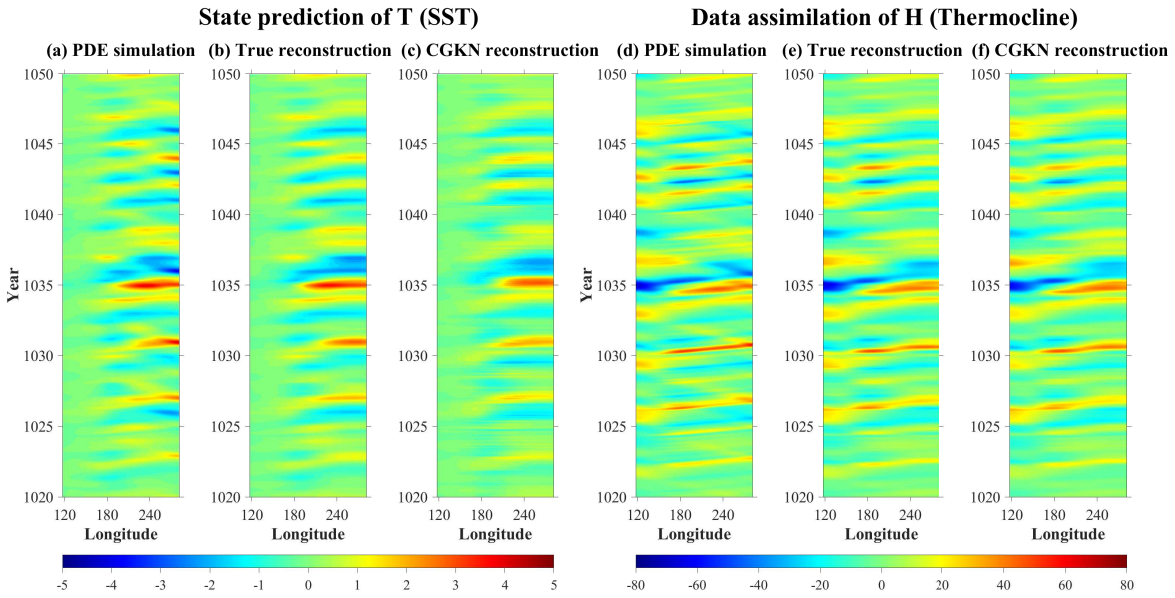

Figure 4.22: Hovmoller diagrams for the simulation from true model and reconstructions by true and approximated states. Panel (a): simulation of **T** (SST) from the true model. Panel (b): reconstruction of **T** from the true discrete states T_W, T_C and T_E which are spatial average of **T**. Panel (c): reconstruction of **T** from predictive states T_W, T_C and T_E of the CGKN model at a 6-month lead time. Panel (d): simulation of **H** (thermocline) from the true model. Panel (e): reconstruction of **H** from the true discrete states H_W, H_C and H_E which are spatial average of **H**. Panel (f): reconstruction of **H** from estimated H_W, H_C, and H_E via DA which apply analytical formulae on CGKN.

The proposed framework integrates the strengths of deep learning methods, Koopman theory, and conditional Gaussian nonlinear systems for modeling complex dynamical systems and efficient DA. In addition, it serves as an effort towards a systematic approach to impose useful structures into deep learning models and to facilitate their usage in other outer-loop applications of scientific machine learning, e.g., control of partially observed systems, and inverse problems. Some extensions and future directions of the proposed framework include more rigorous approaches to efficiently quantify the uncertainties associated with the DA results of the CGKN model and multi-objective optimization methods to enhance the overall performance of the nonlinear transformation, the state predictions, and the DA.

4.3 Discrete-Time Conditional Gaussian Koopman Network

Complex dynamical systems are ubiquitous across a wide range of science and engineering disciplines, including solid mechanics, fluid dynamics, geophysics, neuroscience, and material science [1–6]. Many of those systems are described by partial differential equations (PDEs) or their stochastic extensions and characterized by complex features such as nonlinearity, multiscale dynamics, chaos, turbulence, shock behavior, intermittency, and non-Gaussian statistics [7–12]. Deriving governing equations for these spatiotemporal dynamical systems from first principles is a conventional way of modeling and understanding these systems. The advantages of having explicit governing equations include adherence to physical laws and interpretability. However, deriving the governing equations usually requires a deep understanding of the underlying physics, extensive domain knowledge, and strong prior assumptions. Additionally, computationally expensive numerical simulation is typically necessary to understand and predict most of these systems, which can further increase the burden of downstream applications such as data assimilation (DA), uncertainty quantification (UQ), and decision-making. As a result, data-driven methods, including closure models [13, 14, 371], sparse identification [15, 16], reduced order models [17–19], causality-based models [20–22], and Gaussian processes [23] have become increasingly promising and practical for studying spatiotemporal dynamical systems and carrying out many of the tasks mentioned above.

As one of these data-driven methods, neural-network-based models in deep learning [24–28] have promoted the field of scientific machine learning (SciML) and have shown promise in the study of spatiotemporal dynamical systems with the rapid growth of available data and computing resources [29–33]. Based on the universal approximation theorem [34–36], neural networks can approximate any nonlinear functions and operators. Consequently, they demonstrate significant capability in discovering unknown dynamics and making predictions. For example, as a continuous-in-time version of ResNet [37], the neural ODE [38] is proposed to learn the dynamics of systems with efficient memory usage. Physics-informed neural networks [39], which integrate physical laws into the neural networks, are developed to solve forward and inverse PDE-governed problems. Designed to learn mappings between infinite-dimensional function spaces, neural operators [40–45] have been widely used to approximate solution operators and build spatiotemporal models.

Recently, generative models [46–50] have been utilized to construct stochastic nonlocal models [51], generate spatiotemporal turbulence [52], capture extreme event statistics [53] and learn PDE solutions [54]. Additionally, active learning [55, 56] has been considered for identifying informative data [57, 58], simulating nonlinear systems [59], estimating model parameters [60], and learning model discrepancies [61].

In recent years, leveraging online observational data to improve the performance of data-driven models has become a critical component of reliably deploying those models in real-world applications, which motivates the joint study of SciML and DA. DA integrates observational data with computational models to enhance state estimation [62–65]. It is extensively applied in real-time forecasting, parameter estimation, imputation, and optimal control [66–70]. The governing equations of the underlying dynamics are utilized as computational models for short-term statistical prediction, known as the prior distribution. Through Bayesian inference, the prior distribution is updated with the observational data to form the so-called posterior distribution, which is the solution of DA. To handle general nonlinear dynamical systems, the ensemble DA methods [71–75] that utilize samples to approximate probabilistic distributions have become commonly used techniques. However, the direct application of ensemble DA methods to high-dimensional spatiotemporal systems is often computationally intractable, primarily due to the substantial computational cost incurred by the curse of dimensionality in sampling processes and the numerical simulations of PDEs. Although the adoption of efficient computational models, such as data-driven surrogate models [76–79], reduced order models [80, 81], and stochastic parameterizations [82–85], coupled with various DA techniques like localization [86, 87] and empirical tunings of ensemble sizes and covariances matrix, can mitigate costs and facilitate the implementation of DA, the DA process still demands significant computational resources, and may suffer from sampling and approximation errors. Recently, deep learning has been employed to support DA for spatiotemporal systems [88], assisting the construction of computational models by neural networks that enhance forecast capability within DA [89, 90, 372]. Additionally, statistical variational DA methods have been proposed to generate observations in scenarios where observational data is scarce [91]. Using encoder-decoder networks in latent DA has proven effective in utilizing multi-domain data for high-dimensional systems [92]. Deep learning can also facilitate using data-driven ensembles at a low computational cost to accelerate ensemble DA [93]. Furthermore, end-to-end deep learning methods have been developed to streamline the entire DA process [94, 95].

While these existing methods have facilitated the implementation of DA with affordable computational costs and demonstrated successes in many applications, efficient DA methods for spatiotemporal dynamical systems still require further exploration and development, especially for incorporating DA into the training of SciML models to enhance their performance. Several recent works have started to explore how to incorporate DA into the training of a machine learning model from various perspectives, e.g., using ensemble Kalman filter to enhance the learning of structural errors for non-ergodic systems via derivative-free optimization [13], developing auto-differentiable ensemble Kalman filter [310, 373], exploiting analytical formulae of DA for a conditional Gaussian neural stochastic differential equation [110]. Importantly, developing a unified modeling framework that can handle both efficient state forecast and DA for spatiotemporal dynamical systems, with a low training cost and robust testing results for problems with high-dimensional (or even infinite-dimensional) system states, can significantly strengthen the joint study of SciML models and their data assimilation performance, potentially advancing the reliable deployment of SciML models in real-world applications.

To establish a unified deep learning framework for SciML modeling and efficient DA, the continuous-time conditional Gaussian Koopman network (CGKN) was proposed in [374], aiming to perform efficient state forecast and DA for the modeling and simulation of spatiotemporal dynamical systems. In contrast to the standard usage of Koopman theory [350], which fully linearizes nonlinear dynamical systems [351–355, 375–380], CGKN leverages the Koopman embedding to discover a proper latent representation for the unobserved system states and employs a conditional linear structure to construct the data-driven model architecture. The resulting system is still highly nonlinear. Nevertheless, it leads to a conditional Gaussian system [110, 193, 194, 311, 374], which encompasses numerous nonlinear models across various disciplines with wide-ranging applications in natural science and engineering [257, 315, 358, 381]. One desirable feature of the conditional Gaussian system is that it allows analytical DA formulae and thus avoids the need for ensemble DA methods. Together with a lower-dimensional latent representation for unobserved states, significantly improved efficiency can be achieved for the DA process in both the training and testing phases. In this work, we establish a discrete-time CGKN framework, which avoids numerical integration with small step sizes for the continuous CGKN framework in [374] and thus significantly enhances the efficiency of handling problems with high-dimensional or even infinite-dimensional state variables. The effectiveness and efficiency of the discrete-time

CGKN framework are demonstrated through several PDE-governed problems, including the viscous Burgers' equation, the Kuramoto—Sivashinsky equation, and the 2-D Navier–Stokes equations.

4.3.1 Problem Statement

Considering a spatiotemporal dynamical system including partial differential equations (PDEs) of the general form

$$\partial_t \mathbf{u} = \mathcal{M}(\mathbf{u}), \tag{4.59}$$

where $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^{d_\mathbf{u}}$ is the system state, with spatial variable $\mathbf{x} \in D_\mathbf{x} \subset \mathbb{R}^{d_\mathbf{x}}$ and temporal variable $t \in [0, L_t]$, and $\mathcal{M}$ is a nonlinear operator. When discretized in both spatial and temporal domains, (4.59) can be written as a discrete dynamical system:

$$\mathbf{u}^{n+1} = \mathcal{G}\big(\mathbf{u}^n\big), \tag{4.60}$$

where $\mathbf{u}^n := \mathbf{u}(\cdot, t_n)|_{D_\mathbf{x}^{(M)}} \in \mathbb{R}^d$ denotes the system state variables evaluated at time t_n and M-point spatial discretization $D_\mathbf{x}^{(M)} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M)}\}$ of the bounded domain $D_\mathbf{x}$. $\mathcal{G}$ is the solution map for this discrete system over time interval $t_{n+1} - t_n = \Delta t$. This map $\mathcal{G}$ is also referred to as the state transition map. It should be noted that the temporal resolution of the data, dictated by the choice of Δt, is often larger than the numerical integration time step δt, which is required by the numerical simulation of (4.59) and is typically small to ensure accuracy and stability of the simulation. Therefore, learning and applying the solution map $\mathcal{G}$ at a coarser temporal resolution Δt can significantly reduce the computational cost.

In this work, we focus on the situation with partial observations, which means the online observational data only contain a subset of the system states of the discrete dynamical system (4.60). The observed states and unobserved states are denoted as $\mathbf{u}_1^n \in \mathbb{R}^{d_1}$ and $\mathbf{u}_2^n \in \mathbb{R}^{d_2}$, respectively, with $\mathbf{u}^n = \{\mathbf{u}_1^n, \mathbf{u}_2^n\}$. For instance, $\mathbf{u}_1^n$ and $\mathbf{u}_2^n$ can be interpreted as the spatial partition of $\mathbf{u}^n$. Consequently, the system in (4.60) can be rewritten as follows:

$$\begin{aligned} \mathbf{u}_1^{n+1} &= \mathcal{G}_1\big(\mathbf{u}_1^n, \mathbf{u}_2^n\big), \\ \mathbf{u}_2^{n+1} &= \mathcal{G}_2\big(\mathbf{u}_1^n, \mathbf{u}_2^n\big), \end{aligned} \tag{4.61}$$

where the two maps $\mathcal{G}_1$ and $\mathcal{G}_2$ are often both nonlinear for many real-world applications.

These nonlinear maps pose challenges to data assimilation (DA), e.g., preventing the direct use of the analytical formulae and thus demanding methods developed for general nonlinear problems such as ensemble Kalman filter or even particle filters that involve the evaluation of high-dimensional quadratures [382].

The goal of this work is to learn a surrogate model that embeds a conditional Gaussian structure, which is capable of performing two tasks for systems governed by (4.61): efficient state forecast and DA. The task of DA aims to the conditional distribution $p(\mathbf{u}_2^n|\{\mathbf{u}_1^i\}_{i=0}^n)$, while the task of state forecast is to predict the future states $\mathbf{u}^n$ without assuming the knowledge for the exact initial unobserved states $\mathbf{u}_2^0$.

More specifically, to achieve efficient DA via the analytical formulae from the conditional Gaussian structure, we choose to avoid directly dealing with the nonlinearity of the unobserved state $\mathbf{u}_2$. According to the Koopman theory [350], a nonlinear dynamical system can be linearized by modeling the evolution the observation function of the system states, where the observation function h is in a Hilbert space on the system state $\mathbf{u}^n$ of the discrete system in (4.60). The Koopman operator is defined as an operator $\mathcal{K}$ that evolves the observation function in that Hilbert space, i.e., $\mathcal{K}h := h \circ \mathcal{G}$, where $\circ$ denotes function composition. In practice, it can often be characterized (or approximated) by a finite-dimensional dynamical system in the form $\mathbf{v}(\mathbf{u}^{n+1}) = \mathbf{A}\mathbf{v}(\mathbf{u}^n)$, where $\mathbf{v}$ is a vector function on the space of $\mathbf{u}^n$, $\mathbf{v}(\mathbf{u}^n) \in \mathbb{R}^{d_v}$ is a finite-dimensional vector, and $\mathbf{A} \in \mathbb{R}^{d_v \times d_v}$ is a matrix. Obtaining such a finite-dimensional linear dynamical system for a general nonlinear system in (4.60) involves the identification of the subspace spanned by eigenvectors of the Koopman operator $\mathcal{K}$, which has been studied in [376] via a deep learning perspective. More details on Koopman theory can be found in 4.3.2.

In this work, instead of directly applying the Koopman theory to the nonlinear dynamical system in (4.60), we exploit the linear embedding enabled by the Koopman theory on the partially observed system in (4.61) to obtain a conditional linear system, i.e., the governing equations become conditionally linear with respect to a latent representation $\mathbf{v}$ of the unobserved states $\mathbf{u}_2$, while the nonlinearity of observed states $\mathbf{u}_1$ is preserved. More details of generalized application of Koopman theory can be found in 4.3.2. The resulting modeled system is written as follows:

$$\begin{aligned} \mathbf{u}_1^{n+1} &= \mathbf{F}_1\left(\mathbf{u}_1^n\right) + \mathbf{G}_1\left(\mathbf{u}_1^n\right)\mathbf{v}^n + \boldsymbol{\sigma}_1\boldsymbol{\epsilon}_1^n, \\ \mathbf{v}^{n+1} &= \mathbf{F}_2\left(\mathbf{u}_1^n\right) + \mathbf{G}_2\left(\mathbf{u}_1^n\right)\mathbf{v}^n + \boldsymbol{\sigma}_2\boldsymbol{\epsilon}_2^n, \end{aligned} \tag{4.62}$$

where $\mathbf{v} \in \mathbb{R}^{d_v}$ denotes a latent representation of unobserved states $\mathbf{u}_2$ in (4.61) and is referred to as latent states. In this modeled system, $\mathbf{F}_1$, $\mathbf{G}_1$, $\mathbf{F}_2$, and $\mathbf{G}_2$ are four (potentially nonlinear) maps of observed states $\mathbf{u}_1$. The $\boldsymbol{\epsilon}_1 \in \mathbb{R}^{d_1}$ and $\boldsymbol{\epsilon}_2 \in \mathbb{R}^{d_v}$ are independent Gaussian white noises with $\boldsymbol{\sigma}_1 \in \mathbb{R}^{d_1 \times d_1}$ and $\boldsymbol{\sigma}_2 \in \mathbb{R}^{d_v \times d_v}$ as the noise coefficients. The two noise terms represent model errors that quantify the approximation quality of the surrogate model to the true underlying system. Despite the use of Gaussian white noises, it is worth highlighting that the nonlinearity in the model can still capture highly non-Gaussian features, thereby compensating for model errors arising from various approximations. In this work, the four nonlinear maps $\mathbf{F}_1$, $\mathbf{G}_1$, $\mathbf{F}_2$, and $\mathbf{G}_2$, together with the nonlinear maps between unobserved states $\mathbf{u}_2$ and the latent states $\mathbf{v}$, are parameterized by neural networks and calibrated based on data of the original dynamical system in (4.61). The modeled system in (4.62) belongs to a class of dynamical systems called conditional Gaussian nonlinear system (CGNS), i.e., the conditional distribution $p(\mathbf{v}^n|\{\mathbf{u}_1^i\}_{i=0}^n)$ is Gaussian, which facilitates efficient analytical formulae of DA. Another advantage of the surrogate model is the latent space embedding, which exploits the possible intrinsic lower-dimensional structure of the unobserved states $\mathbf{u}_2$ and thus can further facilitate the computational efficiency of state forecast and DA. A schematic overview of the surrogate model is shown in Figure 4.23(a).

It is worth noting that $\mathbf{v}$ is constructed as a latent representation of $\mathbf{u}_2$, and its interaction with $\mathbf{u}_1$ is facilitated by the conditional Gaussian structure in (4.62), to allow for the use of analytical DA formulae. The modeled system in (4.62) is related to the CGKN framework proposed in [374] and can be viewed as a discrete version of the original CGKN framework. Unlike the continuous form of CGKN framework studied in [374], the discrete form in (4.62) avoids the numerical integration along time with a small step size and thus enables the exploration of its performance on PDE-governed problems with high-dimensional (or even infinite-dimensional) system states. The choice of temporal step size involves a trade-off: a smaller step enables high temporal resolution capture of fine-scale dynamics, whereas a larger step is more suitable for modeling coarse-grained or long-term behaviors. The remainder of this section introduces the architecture of CGKN in its discrete-in-time form, the application of CGKN to state forecast and DA, the training of CGKN based on offline data, and the uncertainty quantification for the results of CGKN.

4.3.2 Generalized Koopman Operator for Discrete-Time System

For nonlinear dynamical systems in the general form:

$$\mathbf{u}^{n+1} = \mathcal{G}(\mathbf{u}^n), \tag{4.63}$$

where $\mathbf{u}^n \in \mathbb{R}^d$ is the system state and $\mathcal{G}$ is a nonlinear map, Koopman theory [350] provides a linear perspective by describing the system in terms of the evolution of observable of the system state. The observable $h : \mathbb{R}^d \mapsto \mathbb{R}$ is a function in the Hilbert space operating on the system state $\mathbf{u}$.

The Koopman operator $\mathcal{K}$ is a linear operator acting on observable function h which is defined as

$$\mathcal{K}h := h \circ \mathcal{G}, \tag{4.64}$$

where $\circ$ denotes function composition.

The evolution of observable h is governed by the Koopman operator:

$$h(\mathbf{u}^{n+1}) = h \circ \mathcal{G}(\mathbf{u}^n) = \mathcal{K}h(\mathbf{u}^n). \tag{4.65}$$

In practice, this infinite-dimensional linear dynamical system and the linear operator $\mathcal{K}$ can be approximated by finite-dimensional representations:

$$\mathbf{v}^{n+1} = \mathbf{A}\mathbf{v}^n + \boldsymbol{\sigma}\boldsymbol{\epsilon}^n, \tag{4.66}$$

where $\mathbf{v}^n \in \mathbb{R}^{d_v}$ is a finite-dimensional vector representing the linear embeddings of the system states $\mathbf{u}^n$, approximating the infinite-dimensional function $h(\mathbf{u}^n)$. The matrix $\mathbf{A} \in \mathbb{R}^{d_v \times d_v}$ is the linear dynamics of this discrete linear dynamical system, which approximates the linear operator $\mathcal{K}$. The $\boldsymbol{\epsilon}^n \in \mathbb{R}^{d_v}$ is Gaussian white noise with $\boldsymbol{\sigma} \in \mathbb{R}^{d_v \times d_v}$ as the noise coefficient.

Considering a partially observed nonlinear dynamical system in the general form of (4.61). The generalized application of Koopman theory to the partially observed nonlinear dynamical system in (4.61) aims to linearize the unobserved states $\mathbf{u}_2^n$ while retaining the nonlinearity of the observed states $\mathbf{u}_1^n$. Therefore, the full system remains highly nonlinear, fundamentally contrasting with the standard Koopman theory. The approximated finite-

dimensional dynamical system is given by:

$$\begin{aligned}\mathbf{u}_1^{n+1} &= \mathbf{F}_1\left(\mathbf{u}_1^n\right) + \mathbf{G}_1\left(\mathbf{u}_1^n\right)\mathbf{v}^n + \boldsymbol{\sigma}_1\boldsymbol{\epsilon}_1^n,\\ \mathbf{v}^{n+1} &= \mathbf{F}_2\left(\mathbf{u}_1^n\right) + \mathbf{G}_2\left(\mathbf{u}_1^n\right)\mathbf{v}^n + \boldsymbol{\sigma}_2\boldsymbol{\epsilon}_2^n,\end{aligned} \tag{4.67}$$

where $\mathbf{v}^n \in \mathbb{R}^{d_v}$ are linear embeddings of unobserved states $\mathbf{u}_2^n$ and $\mathbf{F}_1$, $\mathbf{G}_1$, $\mathbf{F}_2$, $\mathbf{G}_2$ are four nonlinear maps of observed states $\mathbf{u}_1^n$. The $\boldsymbol{\epsilon}_1^n \in \mathbb{R}^{d_1}$ and $\boldsymbol{\epsilon}_2^n \in \mathbb{R}^{d_v}$ are independent Gaussian white noises with $\boldsymbol{\sigma}_1 \in \mathbb{R}^{d_1 \times d_1}$ and $\boldsymbol{\sigma}_2 \in \mathbb{R}^{d_v \times d_v}$ as noise coefficients.

4.3.3 Architecture of Discrete CGKN

As illustrated in Figure 4.23, the CGKN is developed to construct a surrogate model in (4.62) to approximate the discrete dynamical system in (4.61). The CGKN comprises an encoder $\boldsymbol{\varphi}$, a decoder $\boldsymbol{\psi}$, and sub-networks $\boldsymbol{\eta}$ to construct the nonlinear maps $\mathbf{F}_1$, $\mathbf{G}_1$, $\mathbf{F}_2$, and $\mathbf{G}_2$. The encoder $\boldsymbol{\varphi}$ is a nonlinear mapping for transforming the unobserved states $\mathbf{u}_2$ to the latent states $\mathbf{v}$, i.e., $\mathbf{v} = \boldsymbol{\varphi}(\mathbf{u}_2)$, while the decoder $\boldsymbol{\psi}$ is its inverse, i.e., $\mathbf{u}_2 = \boldsymbol{\psi}(\mathbf{v})$.

4.3.3.1 Discrete CGKN for State Forecast

Assuming that the values of current states are known, the modeled system in (4.62) can be used to forecast the future states of spatiotemporal dynamical systems. Here we denote the data of the state variables at time t_n as $\mathbf{u}^{n\star} = \{\mathbf{u}_1^{n\star}, \mathbf{u}_2^{n\star}\}$, where the superscript $\star$ indicates the data from the true system in (4.61). Firstly, the unobserved states $\mathbf{u}_2^{n\star}$ are transformed into latent states $\mathbf{v}^{n\star}$ by the encoder $\boldsymbol{\varphi}$, formalizing the current state of the modeled system $\mathbf{u}_{CG}^{n\star}$:

$$\mathbf{u}_{CG}^{n\star} := \begin{bmatrix}\mathbf{u}_1^{n\star}\\ \mathbf{v}^{n\star}\end{bmatrix} = \begin{bmatrix}\mathbf{u}_1^{n\star}\\ \boldsymbol{\varphi}\left(\mathbf{u}_2^{n\star}\right)\end{bmatrix}. \tag{4.68}$$

The forecast states $\mathbf{u}_{CG}$ after time t_n can then be obtained from (4.62) by solving an initial value problem. It is worth noting that the predictive latent states $\mathbf{v}$ in the forecast states $\mathbf{u}_{CG}$ need to be transformed back to the original unobserved states $\mathbf{u}_2$, which is achieved via the decoder $\boldsymbol{\psi}$.

In practice, the data of the latent states $\mathbf{v}^{n\star}$ is often not available due to the lack of unobserved states data $\mathbf{u}_2^{n\star}$, which accounts for a key motivation of performing DA. Generally speaking, DA is a commonly used technique to leverage the data of observed states $\mathbf{u}_1$ to

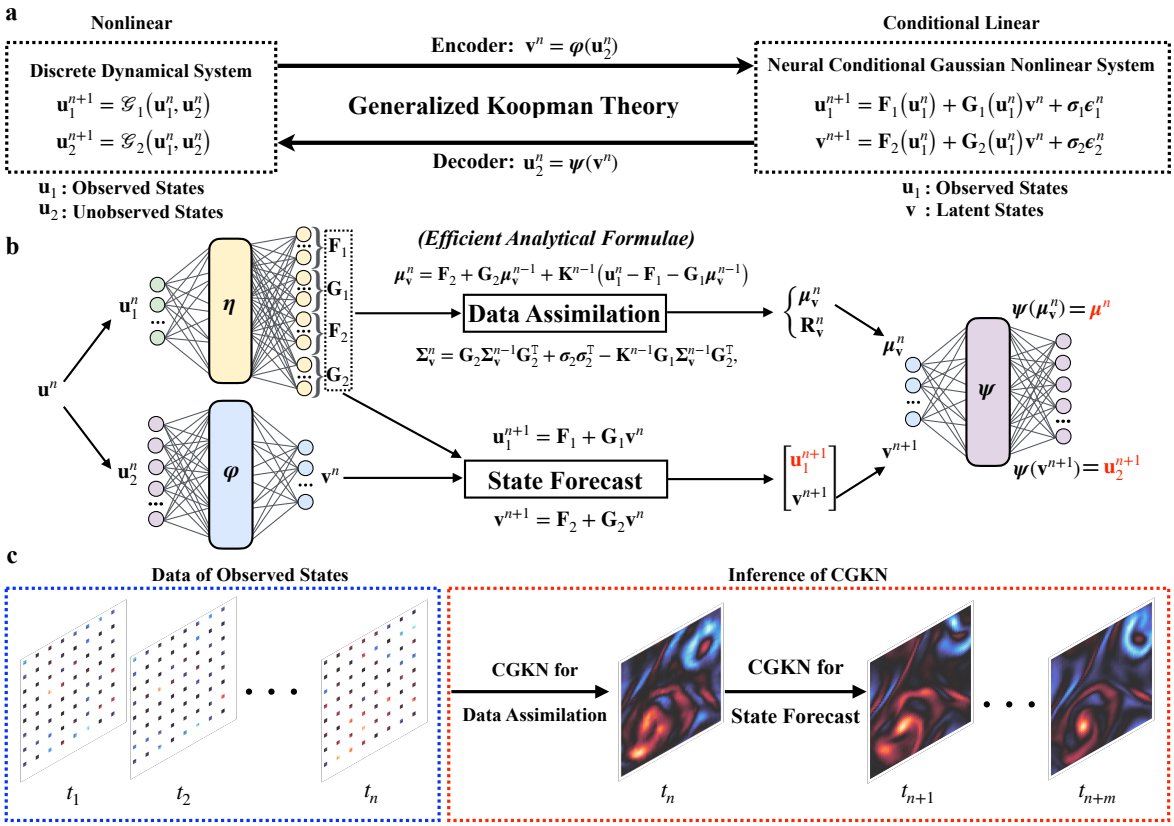

Figure 4.23: Schematic diagram of the architecture and application of the conditional Gaussian Koopman network (CGKN). **a**, Overview of the transformation from discrete dynamical systems to surrogate models following conditional Gaussian structure via generalized application of Koopman theory. **b**, The architecture and workflow of CGKN. The CGKN, consisting of an encoder $\boldsymbol{\varphi}$, a decoder $\boldsymbol{\psi}$ and sub-networks $\boldsymbol{\eta}$, is developed to learn the surrogate model for performing efficient DA via analytical formulae and state forecast for original dynamical systems. **c**, An application of CGKN to Navier–Stokes equations for data assimilation and state forecast.

improve the estimation of $\mathbf{v}$, by obtaining the posterior distribution $p(\mathbf{v}^n|\{\mathbf{u}_1^i\}_{i=0}^n)$, and its maximum posterior point can serve as an estimation of the unknown latent states $\mathbf{v}^{n\star}$. If the Gaussian assumption is adopted in the DA methods, e.g., Kalman filters, the maximum posterior point becomes the mean value $\boldsymbol{\mu}_{\mathbf{v}}$ of the posterior distribution.

Instead of merely employing DA methods (e.g., ensemble Kalman filter) as a separate task to improve the state estimation of nonlinear models, an important contribution of this work is to exploit the conditional linear structure of the modeled system in (4.62) and to adopt the analytical formulae of DA that facilitate an efficient evaluation of the DA performance, which allows the incorporation of the DA performance into the training of a scientific machine learning model and thus enables a unified framework for scientific machine learning and data assimilation. More details of the analytical formulae of DA and the incorporation of DA performance into the model training are discussed in Sections 4.3.3.2 and 4.3.4, respectively.

4.3.3.2 Discrete CGKN for Data Assimilation

A key feature of the CGKN is that the conditional distribution $p(\mathbf{v}^n|\{\mathbf{u}_1^i\}_{i=0}^n)$ is Gaussian, whose mean $\boldsymbol{\mu}_{\mathbf{v}}$ and covariance $\boldsymbol{\Sigma}_{\mathbf{v}}$ can be solved by the analytical formulae:

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{v}}^{n+1} &= \mathbf{F}_2 + \mathbf{G}_2\boldsymbol{\mu}_{\mathbf{v}}^n + \mathbf{K}^n\big(\mathbf{u}_1^{n+1} - \mathbf{F}_1 - \mathbf{G}_1\boldsymbol{\mu}_{\mathbf{v}}^n\big), \\ {}^{\circ}{}_{\mathbf{v}}^{n+1} &= \mathbf{G}_2\,{}^{\circ}{}_{\mathbf{v}}^{n}\mathbf{G}_2^{\mathsf{T}} + \boldsymbol{\sigma}_2\boldsymbol{\sigma}_2^{\mathsf{T}} - \mathbf{K}^n\mathbf{G}_1\,{}^{\circ}{}_{\mathbf{v}}^{n}\mathbf{G}_2^{\mathsf{T}}, \end{aligned} \tag{4.69}$$

where $\mathbf{K}^n = \mathbf{G}_2\,{}^{\circ}{}_{\mathbf{v}}^{n}\mathbf{G}_1^{\mathsf{T}}\big(\boldsymbol{\sigma}_1\boldsymbol{\sigma}_1^{\mathsf{T}} + \mathbf{G}_1\,{}^{\circ}{}_{\mathbf{v}}^{n}\mathbf{G}_1^{\mathsf{T}}\big)^{-1}$ is an analog to the Kalman gain. The mean $\boldsymbol{\mu}_{\mathbf{v}}$ of the latent states $\mathbf{v}$ can be transformed to the posterior mean $\boldsymbol{\mu}$ of unobserved states $\mathbf{u}_2$ by the decoder $\boldsymbol{\psi}$, i.e., $\boldsymbol{\mu} = \boldsymbol{\psi}(\boldsymbol{\mu}_{\mathbf{v}})$. It is worth noting that such a transformation of the estimated mean $\boldsymbol{\mu}_{\mathbf{v}}$ generally does not hold, but the DA loss term used to train the CGKN model can ensure the validity of the posterior mean $\boldsymbol{\mu}$. More discussions about the DA loss term can be found in Section 4.3.4. On the other hand, the posterior covariance $\boldsymbol{\Sigma}$ of the unobserved states $\mathbf{u}_2$ can be estimated through residual analysis, and additional information on the uncertainty quantification can be found in Section 4.3.5.2. It should be noted that the initial condition of (4.69) is typically unknown. Therefore, a warm-up period is required to mitigate initialization discrepancy.

The analytical formulae in (4.69), together with the lower dimensionality of latent state ($d_{\mathbf{v}} \ll d_2$) when an intrinsic lower-dimensional representation of unobserved states is

possible, account for two key features of CGKN that enable efficient DA compared with ensemble DA methods (e.g., ensemble Kalman filter). The computational complexity of CGKN for DA is $\mathcal{O}(d_{\mathbf{v}}^3)$, where $d_{\mathbf{v}}$ is the dimension of the latent states, while that of ensemble DA methods is $\mathcal{O}(Jd_2^2)$, where J is the ensemble size and d_2 is the dimension of unobserved states. To mitigate the sampling errors in ensemble DA methods, a large ensemble size may be needed to well characterize the covariance structure, and thus we often have $Jd_2^2 \gg d_{\mathbf{v}}^3$, making the proposed CGKN framework a more efficient choice for DA than ensemble-based methods. It is worth noting that a small ensemble size J is often employed in practice even for large-scale problems, e.g., numerical weather prediction (NWP) commonly uses an ensemble size on the order of 10^2, despite the dimension of system state being several orders of magnitude larger. The use of a small ensemble size for large-scale problems, together with various techniques such as localization [383, 384] or damping small correlations [385], is still an active research area for ensemble DA methods.

4.3.4 Learning Discrete CGKN from Data

For the training of CGKN model, we assume that offline data $\{\mathbf{u}^{n\star}\}_{n=0}^{N}$ of both observed states $\mathbf{u}_1$ and unobserved ones $\mathbf{u}_2$ are available from the true system in (4.60). The encoder parameters $\boldsymbol{\theta}_{\boldsymbol{\varphi}}$, decoder parameters $\boldsymbol{\theta}_{\boldsymbol{\psi}}$ and sub-networks parameters $\boldsymbol{\theta}_{\boldsymbol{\eta}}$ of the CGKN model illustrated in Figure 4.23(b) are optimized by minimizing a total loss function:

$$\min_{\boldsymbol{\theta}_{\boldsymbol{\varphi}},\boldsymbol{\theta}_{\boldsymbol{\psi}},\boldsymbol{\theta}_{\boldsymbol{\eta}}} L(\boldsymbol{\theta}_{\boldsymbol{\varphi}},\boldsymbol{\theta}_{\boldsymbol{\psi}},\boldsymbol{\theta}_{\boldsymbol{\eta}}), \tag{4.70}$$

which includes autoencoder loss L_{AE}, forecast loss of original states $L_{\mathbf{u}}$, forecast loss of latent states $L_{\mathbf{v}}$, and DA loss L_{DA}:

$$L(\boldsymbol{\theta}_{\boldsymbol{\varphi}},\boldsymbol{\theta}_{\boldsymbol{\psi}},\boldsymbol{\theta}_{\boldsymbol{\eta}}) := \lambda_{AE}L_{AE}(\boldsymbol{\theta}_{\boldsymbol{\varphi}},\boldsymbol{\theta}_{\boldsymbol{\psi}}) + \lambda_{\mathbf{u}}L_{\mathbf{u}}(\boldsymbol{\theta}_{\boldsymbol{\varphi}},\boldsymbol{\theta}_{\boldsymbol{\psi}},\boldsymbol{\theta}_{\boldsymbol{\eta}}) + \lambda_{\mathbf{v}}L_{\mathbf{v}}(\boldsymbol{\theta}_{\boldsymbol{\varphi}},\boldsymbol{\theta}_{\boldsymbol{\eta}}) + \lambda_{DA}L_{DA}(\boldsymbol{\theta}_{\boldsymbol{\psi}},\boldsymbol{\theta}_{\boldsymbol{\eta}}), \tag{4.71}$$

where

$$L_{AE} := \mathbb{E}_{\mathbf{u}_2^\star} \|\mathbf{u}_2^\star - \boldsymbol{\psi}\big(\boldsymbol{\varphi}(\mathbf{u}_2^\star)\big)\|^2, \tag{4.72}$$

$$L_{\mathbf{u}} := \mathbb{E}_{\mathbf{u}^{0\star}} \frac{1}{N_s} \sum_{n=1}^{N_s} \|\mathbf{u}^{n\star} - \mathbf{u}^n\|^2, \tag{4.73}$$

$$L_{\mathbf{v}} := \mathbb{E}_{\mathbf{u}^{0\star}} \frac{1}{N_s} \sum_{n=1}^{N_s} \|\mathbf{v}^{n\star} - \mathbf{v}^n\|^2, \tag{4.74}$$

$$L_{DA} := \mathbb{E}_{\mathbf{u}^{0\star}} \frac{1}{N_l - N_b} \sum_{n=N_b+1}^{N_l} \|\mathbf{u}_2^{n\star} - \boldsymbol{\mu}^n\|^2, \tag{4.75}$$

with λ_{AE}, $\lambda_{\mathbf{u}}$, $\lambda_{\mathbf{v}}$, λ_{DA} being the weights for each loss term. $\|\cdot\|$ denotes the standard vector ℓ^2-norm. The N_s, N_l, and N_b are three hyper-parameters in the training settings which stand for forecast steps, DA steps, and DA warm-up steps, respectively. The $\mathbf{u}^n$ and $\mathbf{v}^n$ are the n-th step state predictions which is described in Section 4.3.3.1, and $\boldsymbol{\mu}^n$ is n-th step DA posterior mean which is detailed in Section 4.3.3.2. In this work, the weights is set as $\lambda_{AE} = 1/d_2$, $\lambda_{\mathbf{u}} = 1/d$, $\lambda_{\mathbf{v}} = 1/d_{\mathbf{v}}$, and $\lambda_{DA} = 1/d_2$, which scale each loss function by a factor of the inverse of the respective vector's dimension. This results in loss functions expressed as the mean squared error (MSE) between the true and approximated data. In practice, the weights can also be treated as hyper-parameters or manually adjusted based on the domain knowledge.

The autoencoder loss L_{AE} in (4.72) is used to minimize the reconstruction discrepancy between the unobserved states $\mathbf{u}_2^{n\star}$ (as the input of autoencoder) and $\boldsymbol{\psi}\big(\boldsymbol{\varphi}(\mathbf{u}_2^{n\star})\big)$ (as the output of autoencoder). The loss L_{AE} promotes the inverse relationship between the encoder and decoder, which helps identify latent states $\mathbf{v} = \boldsymbol{\varphi}(\mathbf{u}_2)$ which can be regarded as embeddings of unobserved states. It is worth noting that the latent space to approximately embed unobserved states is usually not unique, and the joint usage of L_{AE} and other loss terms facilitate the discovery of a proper latent space that is favored by the goals of having good performance in the forecast and DA results of the CGKN model. More specifically, the forecast loss of original states $L_{\mathbf{u}}$ and forecast loss of latent states in $L_{\mathbf{v}}$ in (4.73) and (4.74) are incorporated to ensure that CGKN makes accurate state forecast, while the DA loss L_{DA} in (4.75), made affordable by the efficient DA formulae in (4.69), is critical to ensure the accuracy of DA results from CGKN. It should be noted that the magnitude of $\mathbf{v}$ and $\mathbf{G}_1$ in

(4.62) could be scaled without impacting system states $\mathbf{u}_1$, and a corresponding scaling of the autoencoder can leave system states $\mathbf{u}_2$ unaffected. In this work, such a scale invariance of $\mathbf{v}$ would not lead to any issue, as $\mathbf{v}$ is only a latent representation, and the system states $\mathbf{u}_1$ and $\mathbf{u}_2$ are of actual interest. In practice, the magnitude of $\mathbf{v}$ is also impacted by the choice of $\boldsymbol{\sigma}_2$ in (4.62).

The forecast loss of original states $L_{\mathbf{u}}$ and forecast loss of latent states $L_{\mathbf{v}}$ defined in (4.73) and (4.74) is incorporated to enable CGKN to make accurate state forecast for dynamical systems. From the true states $\{\mathbf{u}^{n\star}\}_{n=0}^{N_s}$ with $\mathbf{u}^{n\star} = \{\mathbf{u}_1^{n\star}, \mathbf{u}_2^{n\star}\}$, the corresponding latent states $\{\mathbf{v}^{n\star}\}_{n=0}^{N_s}$ can be obtained from unobserved states via encoder: $\mathbf{v}^{n\star} = \boldsymbol{\varphi}\big(\mathbf{u}_2^{n\star}\big)$. Based on the initial states $\{\mathbf{u}_1^{0\star}, \mathbf{v}^{0\star}\}$, the predictive states $\{\mathbf{u}_1^n, \mathbf{v}^n\}$ can be obtained by recursively evaluating the surrogate model as illustrated in Section 4.3.3.1. The predictive unobserved states are transformed from the predictive latent states via decoder: $\mathbf{u}_2^n = \boldsymbol{\psi}\big(\mathbf{v}^n\big)$. $L_{\mathbf{u}}$ measures the discrepancy between true states $\{\mathbf{u}^{n\star}\}_{n=0}^{N_s}$ and predictive states $\{\mathbf{u}^n\}_{n=0}^{N_s}$. $L_{\mathbf{v}}$ measures the discrepancy between true latent states $\{\mathbf{v}^{n\star}\}_{n=0}^{N_s}$ and predictive latent sates $\{\mathbf{v}^n\}_{n=0}^{N_s}$. The N_s is the forecast horizon, which can be set as a hyper-parameter in the training setting.

The DA loss L_{DA} in (4.75), which is made affordable by the efficient DA formulae in (4.69), is critical to ensure the accuracy of DA results from CGKN. Based on the data of observed states $\{\mathbf{u}_1^{n\star}\}_{n=0}^{N_l}$, the posterior mean of the latent states $\{\boldsymbol{\mu}_{\mathbf{v}}^n\}_{n=0}^{N_l}$ are calculated by applying the analytical DA formulae in (4.69). The posterior mean of unobserved states $\{\boldsymbol{\mu}^n\}_{n=0}^{N_l}$ is obtained by the decoder: $\boldsymbol{\mu}^n = \boldsymbol{\psi}\big(\boldsymbol{\mu}_{\mathbf{v}}^n\big)$. The DA loss L_{DA} quantifies the discrepancy between the true unobserved states $\{\mathbf{u}_2^{n\star}\}_{n=0}^{N_l}$ and DA posterior mean $\{\boldsymbol{\mu}^n\}_{n=0}^{N_l}$, excluding the first N_b steps. N_l represents the number of DA steps, and N_b denotes the DA warm-up steps, both of which are hyper-parameters in the training setting. The role of DA loss has been investigated in [110].

It should be noted that the analytical DA formulae in (4.69) require the value of $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ in the surrogate model in (4.62). Therefore, prior to incorporating the DA loss into the target loss function to train CGKN, the $\boldsymbol{\sigma}_1$, which is assumed to be a diagonal matrix, is estimated by root mean squared error (RMSE) between the true observed states and one-step predictive observed states via a trained CGKN without DA loss. On the other hand, the $\boldsymbol{\sigma}_2$, which is the noise coefficient of latent states, is on the bias term in the analytical DA formulae in (4.69). In practice, $\boldsymbol{\sigma}_2$ can be set either manually or as a trainable parameter jointly learned with all the trainable maps in the CGKN model. We have studied different choices of $\boldsymbol{\sigma}_2$,

and the results of the learned CGKN models are robust based on those numerical tests. The detailed results are omitted here for simplicity. A brief algorithm about the procedures of learning CGKN from data is summarized in Algorithm 2. More details about the uncertainty quantification are introduced in the following section.

4.3.5 Uncertainty Quantification for Discrete CGKN

This section discusses the uncertainty quantification (UQ) for state forecast and DA results of CGKN. The task of UQ for state forecast focuses on the estimation of $\boldsymbol{\sigma}_1$, which is the noise coefficient of the observed states $\mathbf{u}_1$ and $\boldsymbol{\sigma}_2$, which is the noise coefficient of the latent states $\mathbf{v}$. The task of UQ for DA is to estimate the posterior covariance of the conditional distribution $p(\mathbf{u}_2^n|\{\mathbf{u}_1^i\}_{i=0}^n)$, which is generally non-Gaussian. Both UQ tasks are post-processes of pre-trained CGKN models: the task of UQ for DA is based on the CGKN model trained with the DA loss term, while the task of UQ for state forecast relies on the CGKN model trained without the DA loss term. More details are introduced in the following sections.

4.3.5.1 Uncertainty Quantification for State Forecast

To perform the task of UQ for state forecast, a CGKN model with the total loss in the form of (4.71) while excluding the DA loss term needs to be trained, i.e., the weight of DA loss L_{DA} in the target loss function L in (4.71) is set as $\lambda_{DA} = 0$. Once having the pre-trained CGKN model without DA loss, the noise coefficient $\boldsymbol{\sigma}_1$ of observed states and noise coefficient $\boldsymbol{\sigma}_2$ of latent states are assumed to be diagonal matrices. The diagonal elements are estimated by the root mean squared error (RMSE) between the true states and one-step predictive states of the pre-trained CGKN model:

$$\begin{aligned} \operatorname{diag}(\boldsymbol{\sigma}_1) &= \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^{n\star}-\mathbf{u}_1^n\right)\odot\left(\mathbf{u}_1^{n\star}-\mathbf{u}_1^n\right)}, \\ \operatorname{diag}(\boldsymbol{\sigma}_2) &= \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{v}^{n\star}-\mathbf{v}^n\right)\odot\left(\mathbf{v}^{n\star}-\mathbf{v}^n\right)}, \end{aligned} \tag{4.76}$$

where the notation $\odot$ is element-wise multiplication. Additionally, the estimated $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ in (4.76) can be utilized to calculate the DA loss through analytical DA formulae in (4.69), which facilitates unifying the training of the CGKN model and the tuning of its DA performance based on the total loss in (4.71) with all loss terms included. It is worth noting that an accurate a priori estimation of those noise coefficients may not be necessary to ensure a good tuning of DA performance, since there are other trainable maps in the analytical DA formulae in (4.69). In practice, the noise coefficients can also be empirically set or jointly learned with those trainable maps of the CGKN model.

4.3.5.2 Uncertainty Quantification for Data Assimilation

As introduced in Section 4.3.4, the DA loss term measures the discrepancy between the true unobserved states $\mathbf{u}_2$ and the posterior mean $\boldsymbol{\mu}$, which promotes the performance of DA in terms of mean estimation. However, the posterior covariance of the unobserved states $\mathbf{u}_2$, which quantifies the uncertainty of the associated posterior mean, is still unaddressed. Since true covariance data for the unobserved states does not exist for many real-world applications, especially for those only with experimental data available, it is not feasible to apply a supervised learning framework to estimate the posterior covariance in a similar fashion as being done for the posterior mean.

Therefore, residual analysis is exploited as a post-processing method for a trained CGKN, to efficiently quantify the uncertainties associated with the posterior mean of the unobserved states $\mathbf{u}_2$. Residual analysis is commonly used to evaluate the goodness of a statistical model with respect to observed data. With a trained CGKN model, the DA can be performed for the training data via analytical formulae in (4.69) to obtain the posterior mean $\boldsymbol{\mu}_{\mathbf{v}}$. With the trained decoder ψ, we can further obtain posterior mean $\boldsymbol{\mu}$ of unobserved states $\mathbf{u}_2$. The residual $\mathbf{r}$ is defined as the absolute difference between true data $\mathbf{u}_2^\star$ and its DA posterior mean $\boldsymbol{\mu}$:

$$\mathbf{r} := |\mathbf{u}_2^\star - \boldsymbol{\mu}|,$$

which indicates the desired uncertainties associated with the posterior mean $\boldsymbol{\mu}$ and is assumed to be a function of observed states $\mathbf{u}_1$. This function is then approximated by an auxiliary neural network trained via a standard regression task from the pairwise dataset $\{(\mathbf{u}_1^{n\star}, \mathbf{r}^n)\}_{n=N_b+1}^{N}$, with the first N_b steps as the warm-up period of DA.

With the trained auxiliary neural network for UQ, the residual associated with the

posterior mean can be estimated, using the corresponding observed states as input and evaluating the output of the auxiliary neural network. This residual, derived from the trained auxiliary neural network, can be regarded as the estimated standard deviation of the associated posterior mean, i.e., the standard deviation of diagonal elements of the posterior covariance, based on maximum likelihood estimation under the Gaussian assumption.

Algorithm 2 Learning CGKN from data

Input: $\{\mathbf{u}^{n\star}\}_{n=0}^{N}$ ▷ Training data

$\boldsymbol{\theta}_{\boldsymbol{\varphi}}^{(1)}, \boldsymbol{\theta}_{\boldsymbol{\psi}}^{(1)}, \boldsymbol{\theta}_{\boldsymbol{\eta}}^{(1)} = \arg\min \left\{ \lambda_{\mathrm{AE}} L_{\mathrm{AE}} + \lambda_{\mathbf{u}} L_{\mathbf{u}} + \lambda_{\mathbf{v}} L_{\mathbf{v}} \right\}$ ▷ Train CGKN without DA loss

$\mathrm{diag}(\boldsymbol{\sigma}) = \{\mathrm{diag}(\boldsymbol{\sigma}_1), \mathrm{diag}(\boldsymbol{\sigma}_2)\} = \mathrm{RMSE}(\mathbf{u}_{\mathrm{CG}}^{\star}, \mathbf{u}_{\mathrm{CG}})$ ▷ UQ for state forecast

$\boldsymbol{\theta}_{\boldsymbol{\varphi}}^{(2)}, \boldsymbol{\theta}_{\boldsymbol{\psi}}^{(2)}, \boldsymbol{\theta}_{\boldsymbol{\eta}}^{(2)} = \arg\min \left\{ \lambda_{\mathrm{AE}} L_{\mathrm{AE}} + \lambda_{\mathbf{u}} L_{\mathbf{u}} + \lambda_{\mathbf{v}} L_{\mathbf{v}} + \lambda_{\mathrm{DA}} L_{\mathrm{DA}} \right\}$ ▷ Train CGKN with DA loss

$\boldsymbol{\theta}_{\mathrm{UQ}}^{*} = \arg\min \mathrm{MSE}\left(\mathbf{r}, \mathrm{NN}(\mathbf{u}_1^{\star}; \boldsymbol{\theta}_{\mathrm{UQ}})\right)$ ▷ UQ for DA

Output: $\boldsymbol{\theta}_{\boldsymbol{\varphi}}^{(2)}, \boldsymbol{\theta}_{\boldsymbol{\psi}}^{(2)}, \boldsymbol{\theta}_{\boldsymbol{\eta}}^{(2)}, \boldsymbol{\sigma}, \boldsymbol{\theta}_{\mathrm{UQ}}^{*}$ ▷ Trained parameters

4.3.6 Numerical Experiments

The effectiveness and efficiency of CGKN in state forecast and DA for spatiotemporal dynamical systems are demonstrated through numerical experiments on several PDE-governed canonical examples and benchmarks against other models. The examples include the viscous Burgers' equation, the Kuramoto–Sivashinsky equation, and the Navier–Stokes equations. The methods that have been tested for each system are summarized as follows:

(i) CGKN. It is the deep learning framework proposed in this work, which formalizes the surrogate model in (4.62) to perform state forecast and DA for spatiotemporal dynamical systems.

(ii) Ensemble Kalman filter (EnKF) [71, 86]. EnKF is a classical method for applying DA to nonlinear dynamical systems. The localization strategy and empirical tunings [386, 387] are often used to enhance the EnKF's performance. The comparison of DA results with the CGKN model highlights the accuracy and efficiency of the CGKN model.

(iii) Direct spatial interpolation. Based on the values of observed states across the spatial domain, interpolation can be utilized to estimate the missing values of unobserved

states, recovering the entire fields. It can be viewed as a naive approach to estimate the full-field information based on spatially sparse observations, and it serves as a baseline reference solution for comparing the DA results from CGKN and EnKF.

(iv) Deep neural network (DNN). The model of DNN with fully-connected layers is calibrated to approximate the unknown map $\mathcal{G}$ on the right-hand side in (4.60), and it serves as one of the baseline benchmarks for the performance of state forecast.

(v) Convolutional neural network (CNN) [27, 28]. The unknown map $\mathcal{G}$ on the right-hand side in (4.60) is approximated using a CNN model, which serves as one of the baseline benchmarks for the performance of state forecast.

(vi) Fourier neural operator (FNO). FNO has been demonstrated in [41] with superior performance than classical machine learning methods (e.g., fully-connected networks and CNN) for PDE-governed problems whose system states are spatiotemporal fields. In this work, the unknown map $\mathcal{G}$ on the right-hand side in (4.60) is approximated by an FNO to predict future states. Although it is expected that FNO can outperform all the previous neural-network-based models (including CGKN) for the performance of state forecast, the key advantage of CGKN is its efficient DA.

To compare the performance of the methods above, the mean squared error (MSE) is used to evaluate the performance of state forecast and DA:

$$\text{MSE} := \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\mathbf{x}_m^{n\star} - \mathbf{x}_m^n\right)^2, \tag{4.77}$$

where $\mathbf{x}_m^{n\star}$ is the true value of the m-th variable at the n-th time step, for $m = 1, ..., M$ and $n = 1, ..., N$, and $\mathbf{x}_m^n$ is its corresponding approximated value. The DA error is defined as the MSE between the true unobserved states and their posterior mean from DA, while the forecast error is defined as the MSE between true states and the predictive ones. Both DA error and forecast error are calculated based on test data. Additionally, the forecast error is calculated based on one-step prediction. while the DA error is calculated over the whole time range of the test dataset. The test results of all methods across all examples are summarized in Table 4.9. For a fair comparison, a comparable number of parameters is used in all those deep-learning-based models.

Table 4.9: Test results of all methods in each numerical example. The errors are mean squared errors (MSE) between true values and approximated values. For a fair comparison, a comparable number of parameters is used in different deep learning models.

Examples / Methods	Viscous Burgers Equation		Kuramoto–Sivashinsky Equation		Navier–Stokes Equations	
	Forecast Error	DA Error	Forecast Error	DA Error	Forecast Error	DA Error
CGKN	7.5683e-04	7.5037e-04	1.1042e-02	2.4927e-02	1.9754e+01	6.0940e+01
EnKF	—	5.8125e-04	—	2.4882e-02	—	6.9010e+01
Interpolation	—	1.3514e-02	—	4.3097e-01	—	1.2844e+02
DNN	6.4816e-03	—	4.7332e-02	—	1.0936e+02	—
CNN	2.3727e-03	—	2.6111e-02	—	3.0600e+01	—
FNO	3.9715e-04	—	5.4859e-03	—	1.7129e+01	—

4.3.6.1 Viscous Burgers' Equation: 1-D PDE with Shock Behavior

The 1-D Burgers' equation is a fundamental nonlinear PDE from fluid mechanics, with many applications in various fields such as gas dynamics, traffic flow, and acoustics. It is widely used for studying complex fluid behaviors, particularly notable for modeling shock waves and Burgulence phenomena. The form of the viscous Burgers' equation is:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}, \tag{4.78}$$

where $x \in (0, L_x)$ with periodic boundary conditions, $t \in (0, L_t]$, ν is the viscosity coefficient, and $u(x, 0)$ is a given initial condition. The simulation settings are $L_x = 1, L_t = 2, \nu = 10^{-3}, \Delta x = 1/1024, \Delta t = 10^{-3}$. Under these conditions, the viscous Burgers' equation displays shock behavior due to the formation of steep gradients in the evolution. We perform 1000 simulations as train data and another 100 simulations as test data. The initial conditions $u(x, 0)$ of these simulations are randomly sampled from Gaussian process $\mathcal{N}(0, 625(-\Delta + 25I)^{-2})$ with periodic boundary conditions where Δ is a Laplace operator and I is an identity operator. The resolution of both the train and the test dataset is sub-sampled to $\Delta t = 0.1$ and $\Delta x = 1/64$. Among the 64 states, 4 states which are uniformly distributed across the spatial domain are set as observed, and the rest 60 states are set as unobserved. More specifically, the indices of observed states are 1, 17, 33, 49. The CGKN has been used to learn a surrogate model for i) estimating the 60 unobserved states from the trajectory of the sparse 4 observations via efficient DA and ii) forecasting the future state of the system given any initial state.

To approximate the state transition map of the viscous Burgers' equation from data using the surrogate model in (4.62), the CGKN including an encoder φ, a decoder ψ, and

sub-networks $\boldsymbol{\eta}$ that output $\mathbf{F}_1$, $\mathbf{G}_1$, $\mathbf{F}_2$, $\mathbf{G}_2$ has been constructed. With the dimension of latent states $d_{\mathbf{v}}$ set to 10, the encoder, which takes the unobserved states $\mathbf{u}_2 \in \mathbb{R}^{60}$ as input and output latent states $\mathbf{v} \in \mathbb{R}^{10}$, become $\boldsymbol{\varphi} : \mathbb{R}^{60} \mapsto \mathbb{R}^{10}$, and its inverse, the decoder, become $\boldsymbol{\psi} : \mathbb{R}^{10} \mapsto \mathbb{R}^{60}$. Consequently, the observed states $\mathbf{u}_1 \in \mathbb{R}^4$ and latent states $\mathbf{v}$ formalize the new system states of the surrogate model in (4.62). Furthermore, the nonlinear components $\mathbf{F}_1 \in \mathbb{R}^4$, $\mathbf{G}_1 \in \mathbb{R}^{4\times 10}$, $\mathbf{F}_2 \in \mathbb{R}^{10}$, $\mathbf{G}_2 \in \mathbb{R}^{10\times 10}$ in the surrogate model are output by the sub-networks $\boldsymbol{\eta} : \mathbb{R}^4 \mapsto \mathbb{R}^{128}$, which take observed states $\mathbf{u}_1$ as input. For each component of CGKN, the number of parameters $\boldsymbol{\theta}_{\boldsymbol{\varphi}}$, $\boldsymbol{\theta}_{\boldsymbol{\psi}}$, and $\boldsymbol{\theta}_{\boldsymbol{\eta}}$ are 17066, 17116, 34330, respectively. To learn the CGKN from data, the training parameters are set as follows: state forecast horizon $t_{N_s} = 2$ time units (20 steps), the DA horizon $t_{N_l} = 1000$ time units (1000 steps), and a warm-up period $t_{N_b} = 0.8$ time units (8 steps). The weights in the target loss function in (4.71) are set as $\lambda_{AE} = 1/d_2$, $\lambda_{\mathbf{u}} = 1/d$, $\lambda_{\mathbf{v}} = 1/d_{\mathbf{v}}$, and $\lambda_{DA} = 1/d_2$.

The test results of state forecast and DA from all methods for the viscous Burgers' equation are summarized in Table 4.9 under the column viscous Burgers' equation. The EnKF is applied on the governing equations of viscous Burgers' equation in (4.78) with the prior knowledge of the initial distribution of Gaussian process. The ensemble size of EnKF is set as $J = 100$ and the initial ensembles are randomly sampled from the same Gaussian process that generates the initial conditions for training data. To ensure stability and accuracy in the numerical simulation of the governing equations, a temporal resolution of $\Delta t = 0.01$ has been employed and a spatial resolution of $\Delta x = 1/256$ has been linearly interpolated from the data. Based on the empirical tunings from training data, neither inflation nor localization strategies are needed when applying the EnKF for the test data in this example.

The CGKN can achieve the same level of accuracy in DA as the EnKF, which applies to the true governing equation, and both methods significantly outperform simple interpolation. Regarding time cost, the CGKN requires 0.02 seconds to perform DA for a single simulation (20 steps), compared to 12 seconds for the EnKF. Although these time costs are based on this numerical example and the computing device used in this work, the CGKN is expected to be more computationally efficient in data assimilation than EnKF for a much wider range of problems and computing devices. It should be noted that the numerical simulation for each ensemble member of the viscous Burgers' equation is sequential. While the time cost of EnKF could be reduced through parallel computing, it would require more memory resources. The time usage confirms the speed advantages of CGKN, which are attributed to the efficient

DA formulae, the reduced dimensionality of the latent states, and fast forecast through the approximation of the state transition map. Additionally, as a data-driven method, the DA from CGKN does not require any empirical tunings and any prior knowledge such as the initial distribution of Gaussian process and the governing equation.

As for the performance of state forecast, the DNN is the worst mainly because the fully-connected layer is less effective at leveraging spatial information. The CGKN outperforms the CNN in state forecast and approaches the results achieved by the FNO, which demonstrates the effectiveness of embedding-based learning in CGKN. It is worth highlighting that all general nonlinear neural-network-based models, including DNN, CNN, and FNO, are not able to perform efficient and accurate DA due to their black-box nature and lack of the constraint of DA loss in the training stage. The numerical results demonstrate that the CGKN is effective in both state forecast and efficient DA for the viscous Burger's equation.

Figure 4.24 shows the spatiotemporal plot of the true simulation of the viscous Burgers' equation, DA posterior means from CGKN and EnKF, and the interpolation result. In the results from the true system, the shock behavior is characterized by the horizontal line in the middle part that indicates a dramatic change in the values of the system state within a small spatial distance. Starting with an initial guess, both the flows of CGKN and EnKF can gradually adjust to match the true flow by assimilating observations from the four observed states. In contrast, the pattern of interpolation results deviates significantly from the truth. Additionally, the position of shock behavior can be well-captured by both CGKN and EnKF, while it is not explicit from the interpolation.

The results of the state forecast from various neural-network-based models and the true simulation are shown in Figure 4.25, using three different initial conditions from the test dataset. To evaluate the performance of state forecast of those models, we assume the initial condition is known and then numerically simulate the modeled systems for a certain amount of time. The predictions from DNN and CNN display noticeable deviations from the true spatial functions, while the CGKN and FNO can match the truth well and capture the shock behavior. This figure illustrates the comparable state forecast performance of CGKN and FNO, highlighting CGKN's capability in state forecast. Compared to all the general nonlinear neural-network-based models, the key advantage of CGKN is its capability of efficient DA, which achieves similar performance as applying EnKF to the true governing equations, while demanding much less computational resources in both tuning and performing DA.

In many real-world applications, the initial condition is often unknown or only partially

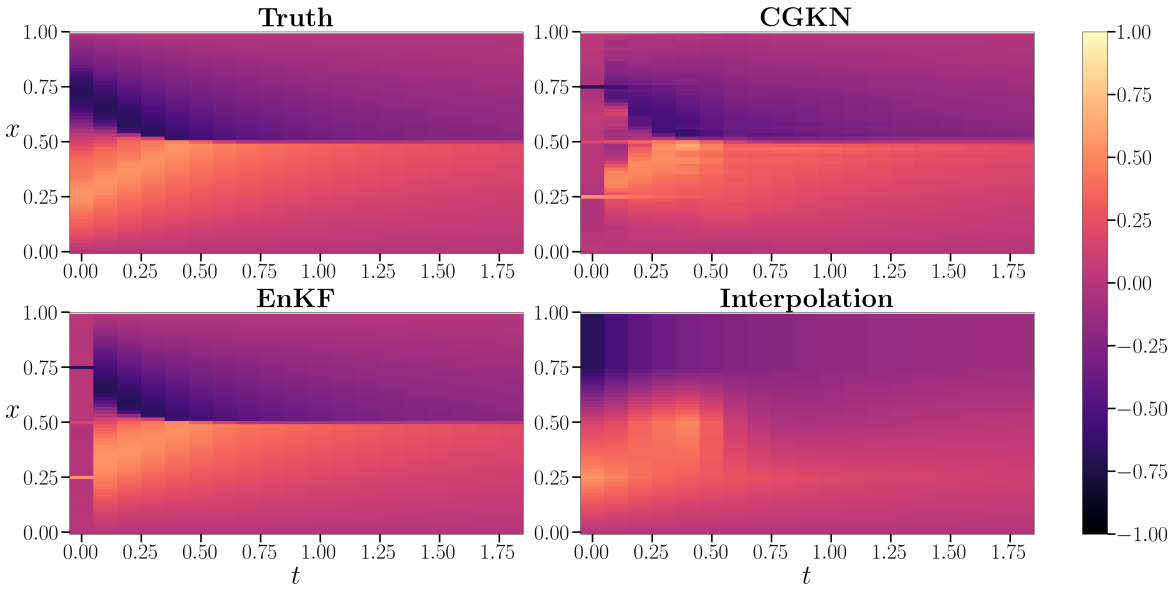

Figure 4.24: Spatiotemporal evolution of the DA results for the viscous Burgers' equation. The spatiotemporal plots of true simulation, DA posterior mean from CGKN, DA posterior mean from EnKF, and interpolation result are shown in each sub-figure. EnKF is applied to the true governing equation with the knowledge of the prior initial distribution.

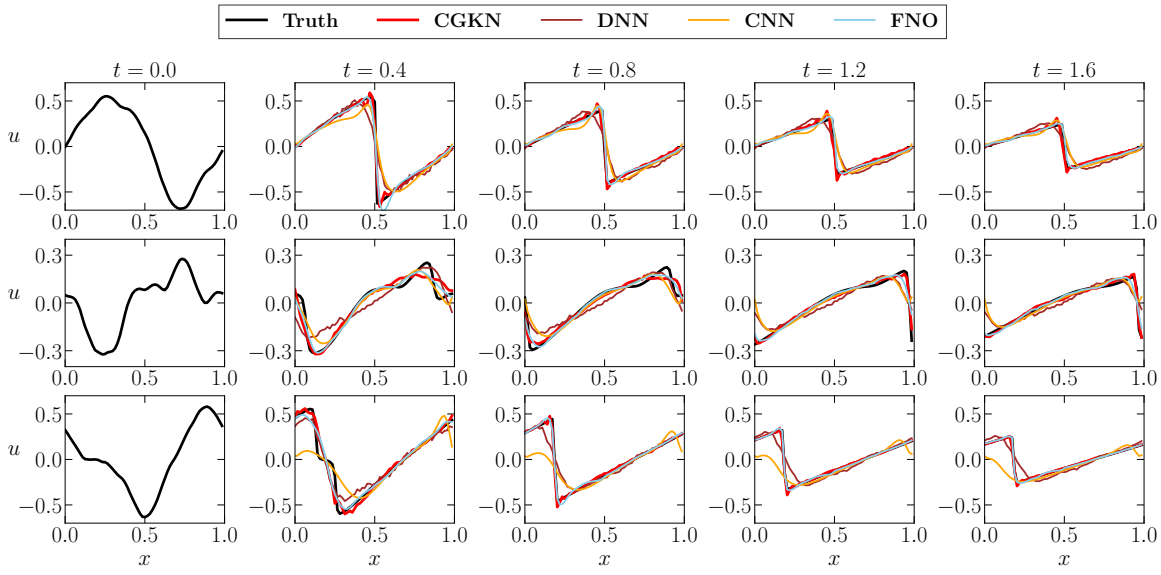

Figure 4.25: Results of state forecast for the viscous Burger's equation. Starting from three different initial conditions in test data, the evolution of true spatial functions is compared with predictive spatial functions from various models, including CGKN, DNN, CNN, and FNO.

known, which accounts for a key motivation of performing DA. Figure 4.26 displays the DA results of unobserved states at the locations $x = 0.375$, 0.624, and 0.875 in the spatial domain. The posterior means with uncertainties characterized by two standard deviations from CGKN and EnKF are compared with the true signals in the first two rows, while the interpolation result is compared with the true signals in the last row. The uncertainty from CGKN is derived from the method of residual analysis, while that from EnKF is obtained from the empirical standard deviation estimated from ensemble samples. Starting from the initial guess, the posterior means from both CGKN and EnKF can match the true signal after about 0.2 time units (2 steps). Though both the uncertainty areas can cover the true signals, the uncertainty area from CGKN is wider than that from EnKF. For the interpolation results, there is a significant deviation from the true signals throughout the entire time range of flow evolution.

Figure 4.27 compares the entire solution profile evolving over time with the DA results obtained from EnKF and CGKN, as well as the interpolation result. It can be seen that a quick adaptation of the DA results to the true signal is achieved via applying EnKF to the true system. At the time $t = 0.2$, the posterior mean from EnKF can almost match the true signal, while there are still some deviations from that of CGKN. The CGKN results gradually align with the true signal in subsequent time steps. It is expected that applying EnKF to the true system can lead to almost perfect state estimation results in subsequent time steps. On the other hand, it is worth mentioning that the posterior mean from CGKN may not always precisely match the true signal at some points, mainly due to the trade-off between fitting a data-driven model and ensuring its generalization capability, especially in regions close to the steep gradient in space. Nevertheless, the uncertainty area still encompasses the true signal for most of those regions, thereby enhancing the reliability of the prediction results.

4.3.6.2 Kuramoto–Sivashinsky Equation: 1-D Chaotic PDE

The Kuramoto—Sivashinsky (K–S) equation is a fourth-order nonlinear PDE that was originally developed to model diffusive-thermal instabilities in a laminar flame front and appears in various areas of physics and engineering such as reaction-diffusion systems. It has been extensively studied for its wave-like dynamics and chaotic behavior, serving as a prototypical example of a system that exhibits spatiotemporal chaos. The form of the

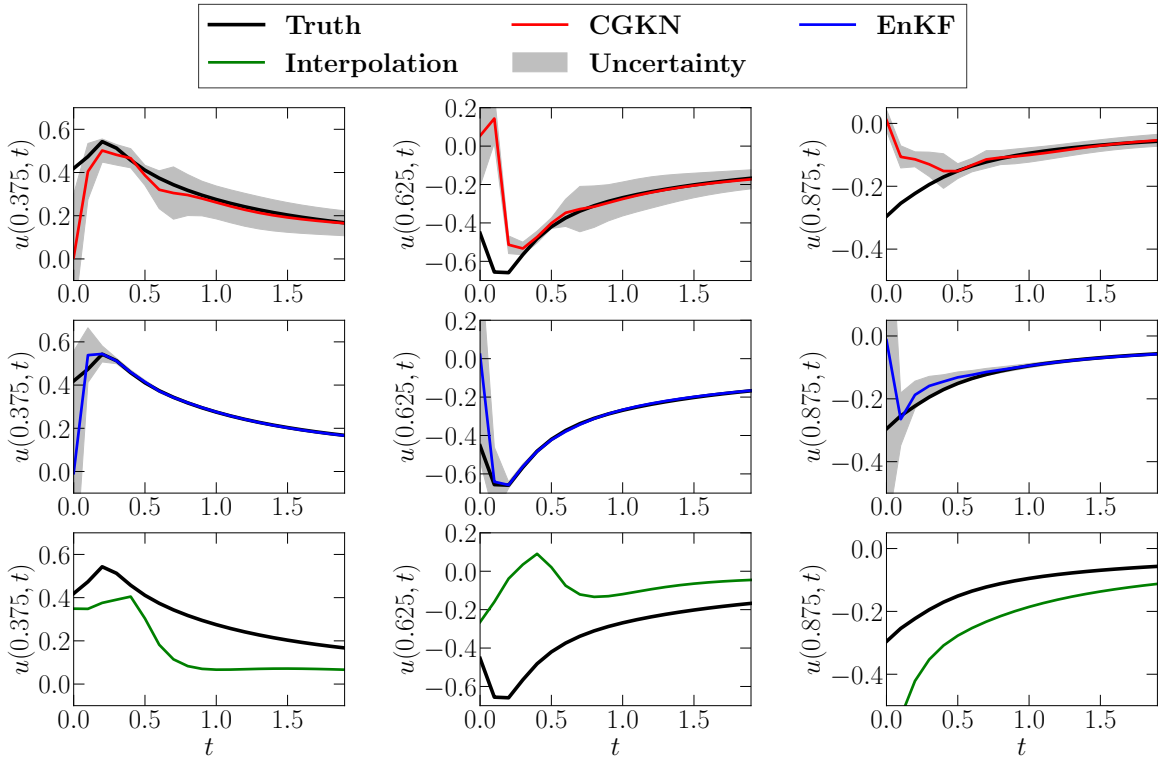

Figure 4.26: Time series of the DA results for the viscous Burgers' equation. True signals, DA posterior means from CGKN and EnKF together with uncertainty areas, and interpolation results for three unobserved states are shown in the figure. The uncertainties of the two standard deviations are indicated by the gray-colored regions associated with the posterior mean.

Kuramoto—Sivashinsky equation is:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}, \tag{4.79}$$

where $x \in (0, L_x)$ with periodic boundary conditions and $t \in (0, L_t]$. The simulation settings are $L_x = 22, L_t = 5000, \Delta x = 22/2048, \Delta t = 0.025$ and the initial condition is $u(x,0) = 0.1 \times \cos(x/16) \times (1 + 2\sin(x/16))$. Under the settings, the K-S equation displays a strong chaotic behavior, making it a good testbed for the proposed method. In the simulation of 5000 time units, the first 80% of the simulation (4000 time units) is used as train data and the remaining 20% (1000 time units) is used as test data. Both train and test data are sub-sampled to resolution $\Delta t = 1$ and $\Delta x = 22/128$. To simulate the real scenarios and demonstrate the

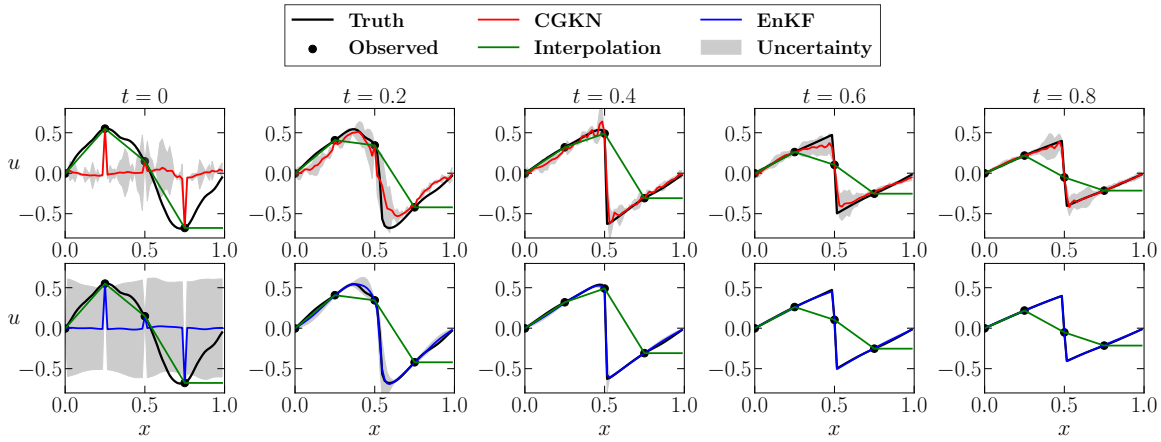

Figure 4.27: Spatial profiles of the DA results for the viscous Burgers' equation. Starting from a random guess, the DA results from CGKN are compared with the true spatial functions in the first row, while those from EnKF are displayed in the second row. The interpolation result is shown in both rows as a reference.

robustness of the proposed method, the observational noise following $\mathcal{N}(0, 0.2^2)$ has been independently and identically added to the true states, including both train and test datasets. The observed states are located at 8 points uniformly distributed across the 1-D spatial domain, whereas the remaining 120 points are unobserved. More specifically, the indices of observed states are 1, 17, 33, 49, 65, 81, 97, 113. The application of CGKN for this 1-D chaotic spatiotemporal system aims to i) estimate the 128 states across the spatial domain from a trajectory data of 8 observed states and ii) forecast the 128 states in the future given any initial 128 states.

With an encoder $\boldsymbol{\varphi}$, a decoder $\boldsymbol{\psi}$, and sub-networks $\boldsymbol{\eta}$ that output $\mathbf{F}_1$, $\mathbf{G}_1$, $\mathbf{F}_2$, $\mathbf{G}_2$, the surrogate model in (4.62) is constructed to approximate the state transition map of the K-S equation based on the data resolution. The dimension of latent state $\mathrm{d}_{\mathbf{v}}$ is selected as 12 in this example. Consequently, the encoder, which take unobserved states $\mathbf{u}_2 \in \mathbb{R}^{120}$ as input and output latent states $\mathbf{v} \in \mathbb{R}^{12}$, becomes $\boldsymbol{\varphi} : \mathbb{R}^{120} \mapsto \mathbb{R}^{12}$ and decoder becomes $\boldsymbol{\psi} : \mathbb{R}^{12} \mapsto \mathbb{R}^{120}$. The observed states $\mathbf{u}_1 \in \mathbb{R}^8$ and latent state $\mathbf{v}$ formalize the system state of the surrogate model in (4.62). The sub-networks $\boldsymbol{\eta} : \mathbb{R}^8 \mapsto \mathbb{R}^{260}$ takes observed states $\mathbf{u}_1$ as input and outputs the nonlinear components $\mathbf{F}_1 \in \mathbb{R}^8$, $\mathbf{G}_1 \in \mathbb{R}^{8\times 12}$, $\mathbf{F}_2 \in \mathbb{R}^{12}$, $\mathbf{G}_2 \in \mathbb{R}^{12\times 12}$ in the surrogate model. The number of parameters $\boldsymbol{\theta}_{\boldsymbol{\varphi}}$, $\boldsymbol{\theta}_{\boldsymbol{\psi}}$, and $\boldsymbol{\theta}_{\boldsymbol{\eta}}$ of each component in CGKN are 18422, 18530, and 37956, respectively. The training settings for the CGKN

include: state forecast horizon $t_{N_s} = \Delta t = 1$ time unit which is the temporal resolution of data (i.e., one-step prediction) and the DA horizon $t_{N_l} = 1000$ time units (1000 steps) with the warm-up period set as $t_{N_b} = 5$ time units (5 steps). The weights in the target loss function are set as $\lambda_{AE} = 1/d_2$, $\lambda_{\mathbf{u}} = 1/d$, $\lambda_{\mathbf{v}} = 1/d_{\mathbf{v}}$, and $\lambda_{DA} = 1/d_2$.

The test results for state forecast and DA from different methods are summarized in Table 4.9 under the column Kuramoto–Sivashinsky equation. The DA error is based on 1000-time-unit (1000 steps) horizon, and the forecast error is calculated by one-step prediction. The interpolation serves as a basic reference model for the DA results from CGKN and EnKF. The EnKF is applied to the true governing equation of K-S equation in (4.79). The ensemble size of EnKF is set as $J = 100$ with the initial ensemble randomly sampled from the training data, in contrast to the initial distribution of Gaussian noise used by DA of CGKN. The constant multiplicative covariance inflation [386] and the localization strategy [387] are used to mitigate sampling errors. Based on the empirical tunings for training data, the inflation and localization parameters are set as 1.05 and 16. The numerical simulation of the governing equations for the forecast step in EnKF employs a smaller temporal resolution of $\Delta t = 0.025$ than the data temporal resolution to ensure numerical stability and accuracy.

For the DA performance, the CGKN is comparable to applying EnKF on the true governing equations, and both methods are significantly superior to the interpolation. Based on the same computing device, the CGKN completes 1000 steps of DA for test data in about 0.8 seconds, while EnKF requires about 100 seconds to accomplish the same task. Although the EnKF can be accelerated via parallel computing for the numerical simulation of ensemble members, it will cost more memory resources. The efficiency of DA from CGKN compared to EnKF stems from the analytical DA formulae, dimension reduction of the latent state, and fast forecast of the approximated state transition map. Therefore, even though the DA performance of CGKN is comparable to that of EnKF, it can significantly reduce computational costs or require fewer computing resources.

For the state forecast, the FNO outperforms all other models and the DNN performs the worst. The performance of CGKN is better than that of CNN and is only slightly less effective than the FNO, a neural-operator-based architecture specifically designed for learning mappings between infinite-dimensional spaces. It should be emphasized that, due to their black-box nature and the absence of DA loss in the training stage, general nonlinear neural network models—including DNNs, CNNs, and FNOs—are not specifically designed for efficient and accurate DA tasks. This highlights the advantage of CGKN.

Figure 4.28 displays the spatiotemporal plot of the true simulation of the K-S equation, posterior DA means from CGKN and EnKF, and interpolation result for 500 time units. The result from CGKN closely resembles that of EnKF, with both methods aligning well with the truth. Both DA methods can effectively capture chaotic patterns and wave-like behaviors. Conversely, the result from interpolation is noticeably different from the true values, although it does manage to qualitatively recover some wave-like patterns.

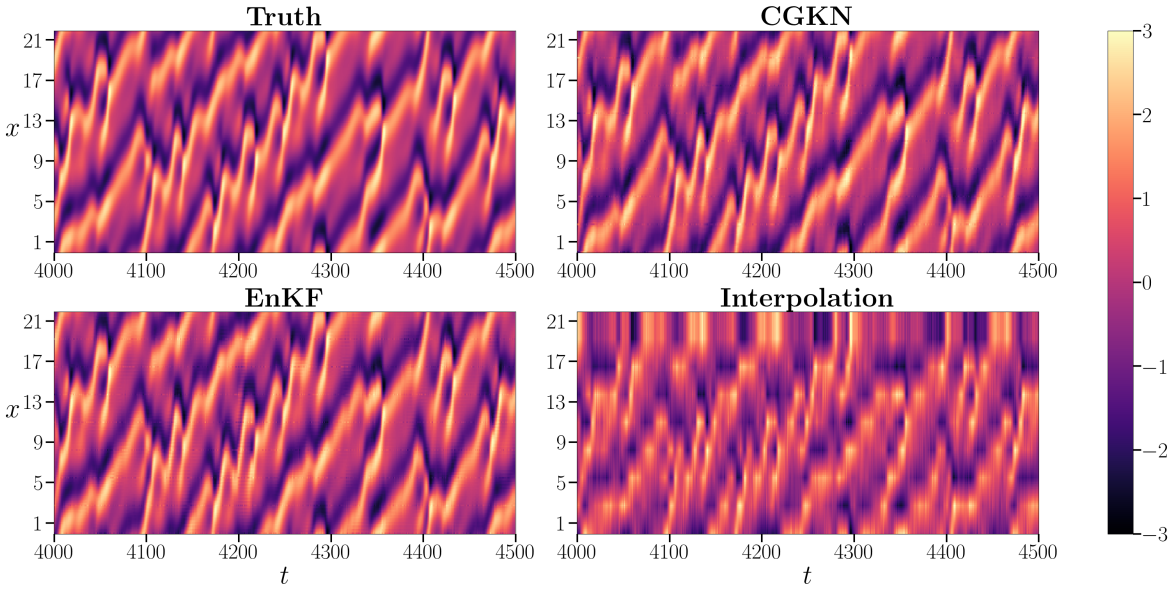

Figure 4.28: Spatiotemporal evolution of the DA results for the Kuramoto–Sivashinsky equation. The spatiotemporal plots of true simulation, DA posterior mean from CGKN, DA posterior mean from EnKF, and interpolation result are displayed in each sub-figure.

Figure 4.29 illustrates the state forecast performance of various models by displaying the evolution of spatial functions of both the true and predictive results starting at 4000 time units, which is in the test dataset. For the one-step prediction, all models accurately match the true spatial function. However, for larger forecast steps, all models gradually diverge from the truth, with CGKN maintaining the closest approximation over the longest forecast horizon in this example. The comparable state forecast capability of CGKN to FNO stems from their similar modeling frameworks: CGKN models the state transition map in latent space, while FNO models that in Fourier space. The effectiveness of CGKN in state forecast primarily benefits from the informative embeddings it learns from data, a feature that is absent in the architectures of DNN and CNN. Similar to the previous numerical example, the key advantage of CGKN over those general nonlinear neural-network-based

models is efficient DA, which achieves comparable performance as applying EnKF to the true governing equations but demands significantly fewer computational resources and does not require empirical tunings.

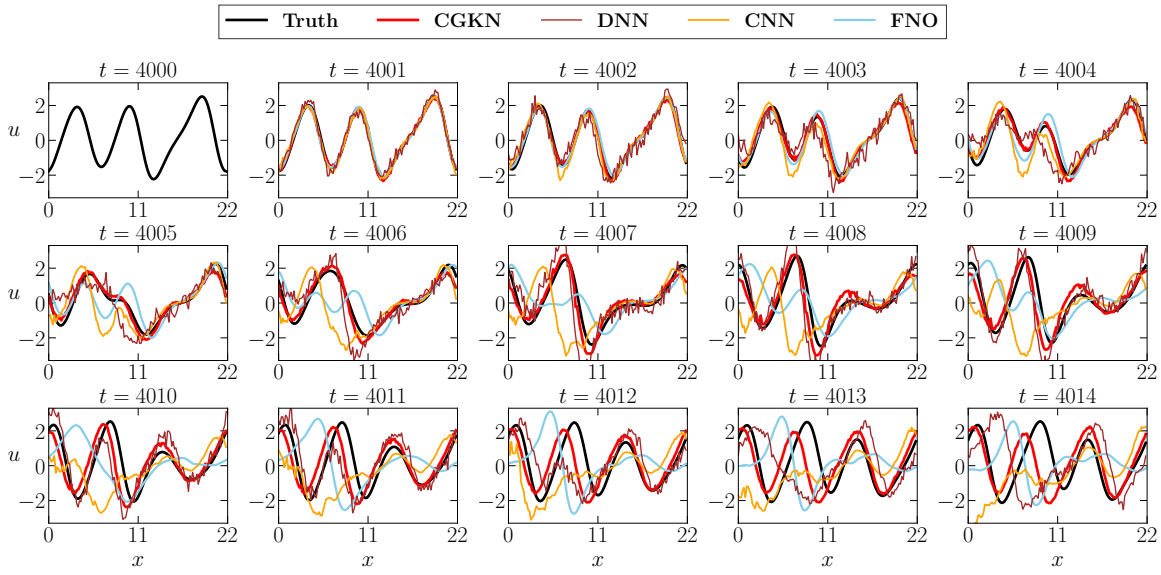

Figure 4.29: Results of state forecast for the Kuramoto–Sivashinsky equation. The evolution of true spatial functions is compared with predictive spatial functions from various models including CGKN, DNN, CNN, and FNO.

Figure 4.30 presents the time series of the unobserved state at the location $x = 9.625$ in the spatial domain $[0, 22]$. The true signal of the unobserved state is compared with the DA posterior means obtained from CGKN and EnKF, as well as with the interpolation result. For both DA methods, the uncertainty area corresponding to two standard deviations is shown around the respective posterior means. The uncertainty from CGKN is derived through residual analysis, while that from EnKF is the empirical standard deviation estimated from ensemble samples. The trajectories of the DA means from both CGKN and EnKF closely match the true signal, significantly outperforming the interpolation results. The uncertainty area provided by CGKN is similar to that of EnKF and covers most of the true signals. Additionally, the DA results from both CGKN and EnKF rapidly adjust to the system and accurately estimate the unobserved state from the initial guess, indicating that only a short warm-up time is required for this spatiotemporal chaotic system.

The initial phases of the DA processes from CGKN and EnKF are illustrated in Figure 4.31, which also includes the true spatial functions and interpolation results. The figure shows the

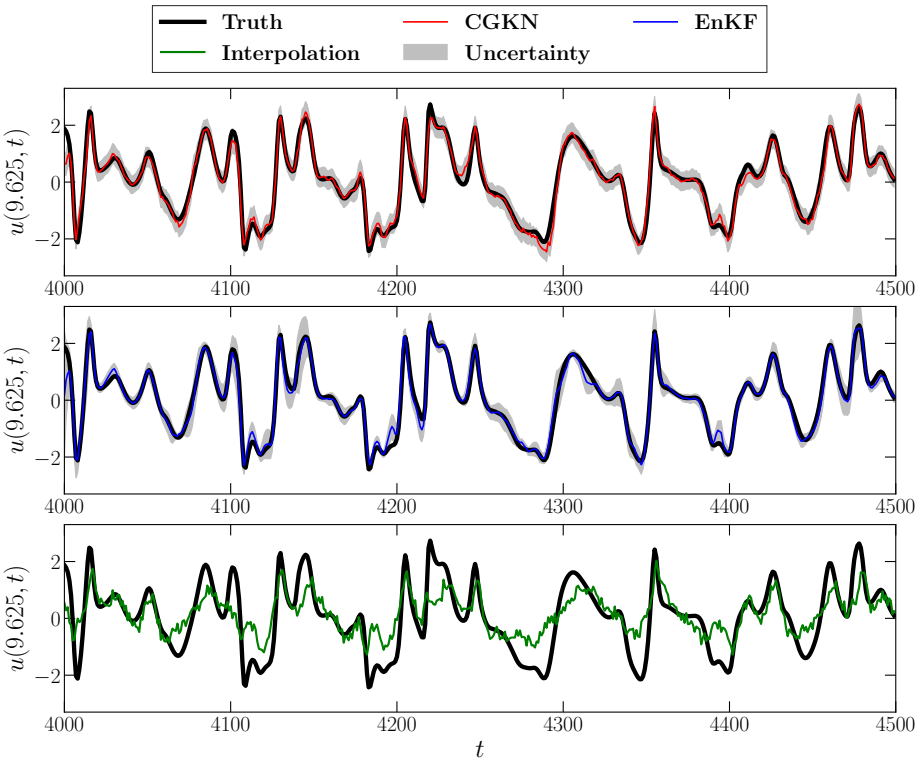

Figure 4.30: Time series of the DA results for the Kuramoto–Sivashinsky equation. True signal, DA posterior mean from CGKN and EnKF together with uncertainty area, and interpolation results for the unobserved state at spatial position 9.625 are presented. The uncertainties of the two standard deviations are indicated by the gray-colored regions associated with the posterior mean.

spatial profiles corresponding to the spatiotemporal evolution presented in Figure 4.28 for the first 10 time units. It is evident that both posterior means align well with the true spatial functions after approximately 6 time units, whereas the interpolation result continues to serve only as a baseline reference. Both Figures 4.30 and 4.31 confirm that CGKN achieves DA performance comparable to using EnKF with the true system.

4.3.6.3 Navier–Stokes Equations: 2-D Turbulent PDE

The Navier–Stokes (N-S) equations are a set of nonlinear PDEs that describe the motion of fluids. These equations are fundamental in the field of fluid dynamics and are used in a wide range of applications in both natural and engineering, including weather patterns, ocean currents, and airflow. The velocity form of the Navier–Stokes equations for incompressible

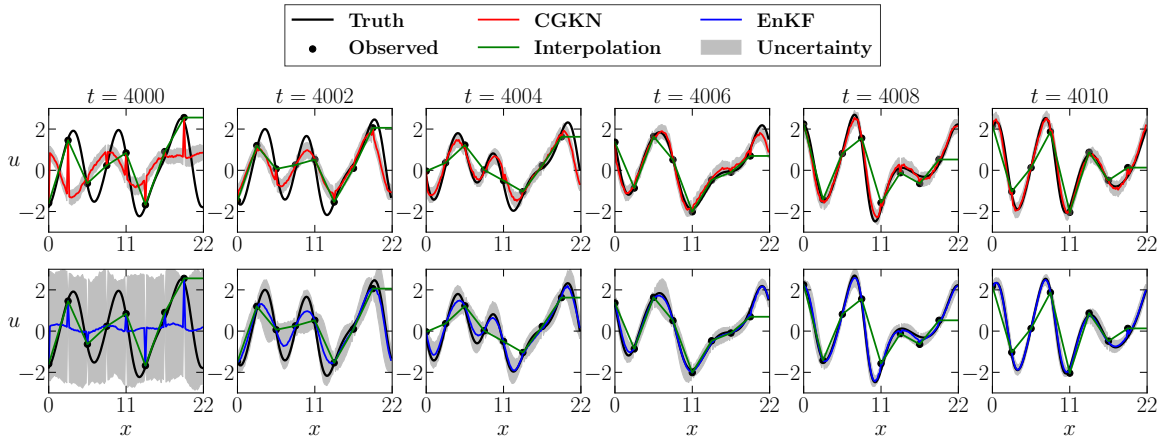

Figure 4.31: Spatial profiles of the DA results for the Kuramoto–Sivashinsky equation. Starting from a random guess, the DA results from CGKN are compared with the true spatial functions in the first row, while those from EnKF are presented in the second row. The interpolation results are provided as references in both rows.

fluid is:

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u}\cdot\nabla\mathbf{u} + \nu\nabla^2\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}, \qquad \nabla\cdot\mathbf{u} = 0, \tag{4.80}$$

where $\mathbf{u}(\mathbf{x},t) \in \mathbb{R}^2$ is a 2-D velocity vector defined over the spatial domain $\mathbf{x} \in \Omega \subset \mathbb{R}^2$ and temporal domain $t \in (0, L_t]$. Here, ν is the kinematic viscosity, ρ is the density, p is the pressure field, and $\mathbf{f}$ is the forcing function. The state of interest in this example is the vorticity, which is the curl of velocity $\omega := \nabla \times \mathbf{u}$. The simulations settings for the system are $\nu = 10^{-4}$, $\rho = 1$, $\mathbf{f} = [100\sin(8y), 0]^T$, $\Omega = (0,1)^2$, $L_t = 1000$, $\Delta t = 10^{-3}$, and $\Delta x = \Delta y = 1/256$. The initial condition is $\mathbf{u}(\mathbf{x},0) = \mathbf{0}$ and the boundary condition is periodic. The N-S equations with this setup result in a 2-D spatiotemporal turbulence. The stable fluids algorithm is used to solve the 2-D Navier–Stokes equations with the above settings to generate 1000-time-unit vorticity. The first 800 units are used as train data and the remaining 200 units are used as test data. To demonstrate the robustness of CGKN and simulate real-world applications, the measurement noise of $\mathcal{N}(0, 2.5^2)$ has been independently and identically added to both the train and test data. The resolution of both train and test data is sub-sampled to $\Delta t = 0.01$, $\Delta x = 1/64$ and $\Delta y = 1/64$. In the 64×64 2-D vorticity field, 8×8 observed points are set uniformly distributed across the spatial domain, while the other points are unobserved. The proposed CGKN for this 2-D turbulent

vorticity field aims to i) estimate the 64×64 vorticity field from a trajectory of 8×8 sparse observed data and ii) forecast the future 64×64 vorticity field given any initial 64×64 vorticity field.

With the CGKN that includes an encoder $\boldsymbol{\varphi}$, a decoder $\boldsymbol{\psi}$, and sub-networks $\boldsymbol{\eta}$ that output $\mathbf{F}_1$, $\mathbf{G}_1$, $\mathbf{F}_2$, $\mathbf{G}_2$, the surrogate model in (4.62) is constructed to approximate the state transition map of the 2-D vorticity field. The hyper-parameter $d_{\mathbf{v}}$, which is the dimension of latent states, has been selected as $16 \times 16 = 256$ in this example. To facilitate the generalized application of Koopman theory for this 2-D turbulence, the convolutional autoencoder has been employed with encoder $\boldsymbol{\varphi} : \mathbb{R}^{64\times64} \mapsto \mathbb{R}^{16\times16}$ and decoder $\boldsymbol{\psi} : \mathbb{R}^{16\times16} \mapsto \mathbb{R}^{64\times64}$. The states in the surrogate model are constituted by observed states $\mathbf{u}_1 \in \mathbb{R}^{64}$ and latent states $\mathbf{v} \in \mathbb{R}^{256}$. The sub-networks $\boldsymbol{\eta} : \mathbb{R}^{64} \mapsto \mathbb{R}^{82240}$ takes observed states $\mathbf{u}_1$ as input and outputs the nonlinear components $\mathbf{F}_1 \in \mathbb{R}^{64}$, $\mathbf{G}_1 \in \mathbb{R}^{64\times256}$, $\mathbf{F}_2 \in \mathbb{R}^{256}$, $\mathbf{G}_2 \in \mathbb{R}^{256\times256}$ in the surrogate model. The number of parameters $\boldsymbol{\theta}_{\boldsymbol{\varphi}}$, $\boldsymbol{\theta}_{\boldsymbol{\psi}}$, and $\boldsymbol{\theta}_{\boldsymbol{\eta}}$ in the CGKN are 187265,187265 and 82240, respectively. The training settings for the CGKN include: state forecast horizon $t_{N_s} = \Delta t = 0.01$ time units which is the temporal resolution of data (i.e., one-step prediction) and the DA horizon $t_{N_l} = 20$ time units (2000 steps) with the warm-up period set as $t_{N_b} = 5$ time units (500 steps). The weights in the target loss function are set as $\lambda_{AE} = 1/d_2$, $\lambda_{\mathbf{u}} = 1/d$, $\lambda_{\mathbf{v}} = 1/d_{\mathbf{v}}$, and $\lambda_{DA} = 1/d_2$.

The test results of state forecast and DA from different methods are summarized in Table 4.9 under the column Navier–Stokes equations. The DA error is based on a 200-time-unit (20000 steps) horizon and the forecast error is calculated by one-step prediction. The EnKF is applied on the true governing equations in (4.80) with the ensemble size set as $J = 100$. Initial conditions of ensembles in EnKF are randomly sampled from the training data, while the initial distribution of CGKN is set as Gaussian white noise. The constant multiplicative covariance inflation and the localization strategy have been taken for EnKF to exploit spatial information and reduce the sampling errors. The parameters of inflation and localization are chosen as 1.1 and 8 based on the empirical tunings of training data. The numerical simulation of the governing equation for the forecast step in EnKF uses a temporal resolution of $\Delta t = 0.001$, compared with 0.01 of the data, to ensure numerical stability and accuracy.

In this 2-D turbulence example, the DA performance of CGKN outperforms that of EnKF, with the interpolation result displaying an error twice as large as that observed in CGKN. Additionally, CGKN completes DA for 20000 steps of test data in about 80 seconds, whereas

EnKF requires about 25000 seconds (about 7 hours) to perform the same task. Though the EnKF can be accelerated by parallel computing for numerical simulation of ensemble numbers, it will cost more memory resources, especially for this 2-D turbulence simulation. The efficient DA of CGKN stems from the availability of analytical DA formulae, the reduced dimensionality of latent state, and fast forecast through approximation of the state transition map. Overall, with substantial advantages in terms of computational cost, the DA result from CGKN is still comparable to that of EnKF.

For state forecast, CGKN achieves performance comparable to FNO and outperforms both CNN and DNN, with DNN delivering the worst results. This is primarily because the DNN fails to leverage spatial information, which is crucial for analyzing a 2-D field. The state forecast performance demonstrates the effectiveness of the embeddings-based learning in the CGKN, which is analogous to the Fourier-modes-based learning in the FNO. It is worth highlighting that due to the black-box nature and lack of the DA loss constraint in the training state, the general nonlinear neural-network-based models including DNN, CNN, and FNO are incapable of performing efficient and accurate DA, which is a key advantage of CGKN.

The state forecast results from different models are presented in Figure 4.32, with the initial condition set as the vorticity at 950 time units from the test dataset. The forecast results from CGKN and FNO successfully capture the flow pattern of the true simulation, whereas the result from CNN deviates significantly, and that from DNN performs the worst. In addition to its superior state forecast performance, a major advantage of CGKN is its capability for efficient and accurate DA.

Figure 4.33 shows the DA results from CGKN and EnKF. Given a trajectory of 8×8 observed states uniformly distributed on the vorticity field starting from 800 time units in test data, the evolutions of the full fields (64×64) of the true simulation, posterior means from CGKN and EnKF, and interpolation result are shown in each row of the figure. The flow patterns of estimated fields from CGKN and EnKF can capture that of the true fields, while the flow pattern of interpolation is very blurry and deviates significantly from the truth. The figure confirms the comparable DA result between CGKN and EnKF, despite CGKN using much fewer computing resources.

Figure 4.34 shows the time series of the unobserved state indexed at the location $(x, y) = (0.5625, 0.5625)$ in the spatial domain of $[0, 1]^2$. The first two rows of the figure present a comparison between the true signal and the DA posterior means from CGKN and EnKF, along

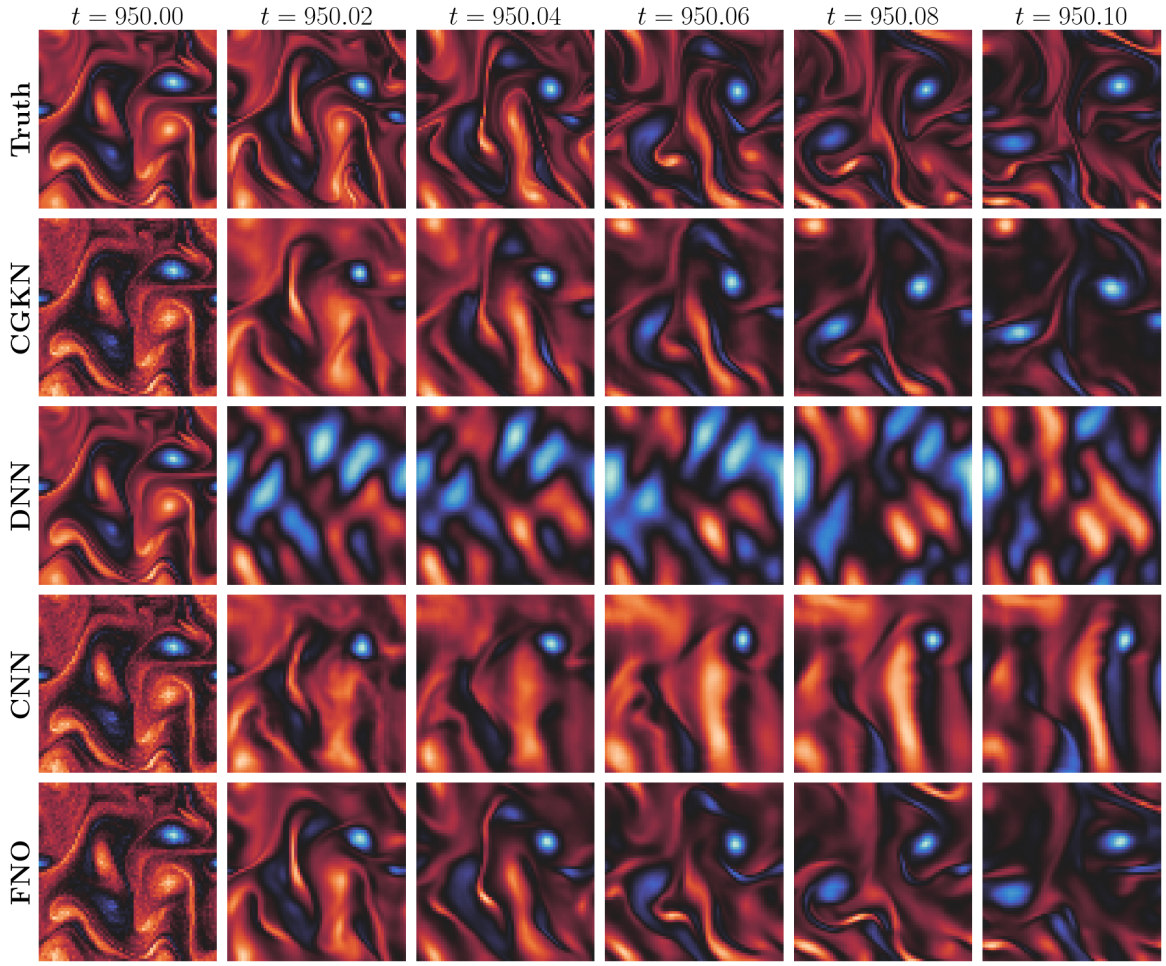

Figure 4.32: Results of state forecast for the Navier–Stokes equations. The evolution of the true vorticity field is compared with the predictive vorticity field from various models including CGKN, DNN, CNN, and FNO.

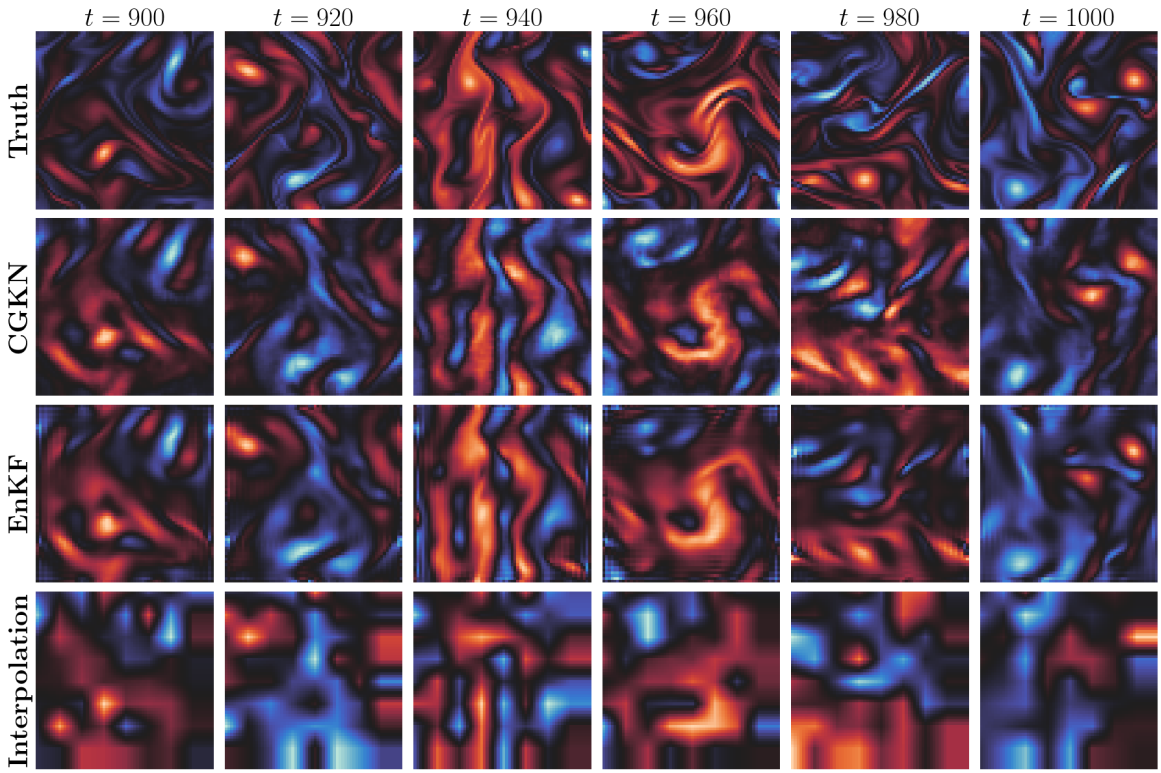

Figure 4.33: Spatiotemporal evolution of the DA results for the Navier–Stokes equations. Heatmaps of each row display the vorticity fields for the true simulation, the DA posterior means from CGKN and EnKF, and the interpolation result.

with their associated uncertainty areas. The uncertainty area is two standard deviations associated with the posterior means, which for CGKN is determined through residual analysis, and for EnKF, it is estimated using the ensemble standard deviation. The interpolation result is shown in the last row. By comparing the posterior mean and the true signal, both CGKN and EnKF outperform the interpolation result, with CGKN demonstrating slightly better performance than EnKF, particularly for some extreme values. Additionally, the uncertainty areas from both CGKN and EnKF can effectively encompass most of the true signals, demonstrating a robust quantification of the uncertainty associated with the estimated mean field. Overall, the CGKN is capable of tackling both tasks of state forecast and efficient DA for this 2-D turbulence example. It is worth noting that the CGKN can achieve comparable DA performance to that of EnKF but with significantly lower computational costs. For state forecast, the CGKN can reach a level of performance similar to that of FNO.

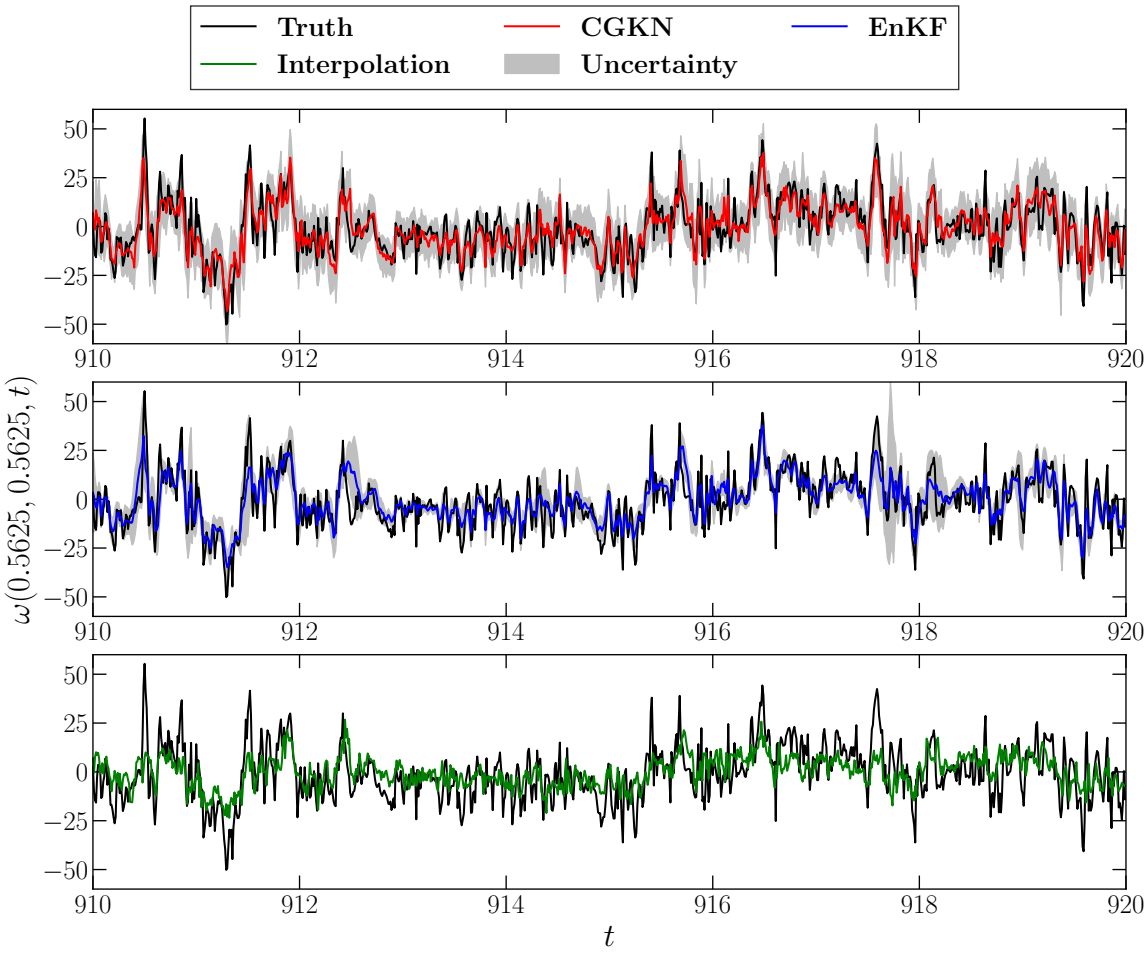

Figure 4.34: Time series of the DA results for the Navier–Stokes equations. The true signal, DA posterior means from CGKN and EnKF together with uncertainty areas of two standard deviations, and interpolation result for the unobserved state at spatial position $(0.5625, 0.5625)$ are presented.

4.3.7 Discussion and Conclusion

This work develops a unified deep learning framework, the discrete-time conditional Gaussian Koopman network (CGKN), to learn a surrogate model to perform efficient data assimilation (DA) and state forecast for spatiotemporal dynamical systems governed by nonlinear partial differential equations. Exploiting the Koopman embedding of the unobserved system states to discover a latent representation with conditional linear dynamics, the spatiotemporal dynamical system can be converted into a conditional Gaussian nonlinear system, serving as a surrogate model of the true system and facilitating analytical formulae for DA. The analytical DA formulae and the lower-dimensional embedded latent space can significantly improve the efficiency of DA, enabling the incorporation of DA performance into the training process of the surrogate model. Therefore, the proposed discrete-time CGKN framework in this work unifies the processes of training a scientific machine learning (SciML) model and tuning its DA performance. The effectiveness of the proposed framework for efficient state forecast and DA is demonstrated through several canonical problems governed by PDEs, including the viscous Burgers' equation, the Kuramoto—Sivashinsky equation, and the 2-D Navier–Stokes equations. Extensions and future directions of the proposed framework CGKN include introducing sparsity or locality in the identified latent states to employ the spatial information for further accelerating the speed of state forecast and DA, exploring alternative approaches to quantifying the uncertainties associated with the posterior mean of CGKN, and employing advanced multi-objective optimization methods to minimize each loss functions for enhancing the training performance of CGKN. Incorporating some known physics and designing more structures of CGKN, e.g., allowing temporal memories to account for non-Markovian dynamics and characterizing the stochastic terms via fluctuation–dissipation theorem, are also interesting future directions. It is worth noting that the proposed CGKN framework also serves as an example to illustrate unifying the development of SciML models and their outer-loop applications, which can potentially inspire research directions of other outer-loop applications such as design optimization, inverse problems, and optimal control.

Chapter 5

Summary

This thesis aims to jointly investigate scientific machine learning (SciML) and data assimilation (DA) to develop and apply reliable and efficient computational methods and algorithms for modeling, simulating, and analyzing complex dynamical systems. Such systems are ubiquitous in many scientific and engineering fields, such as fluid dynamics, geophysics, and materials science, and often exhibit complex behaviors and phenomena, including strong nonlinearity, high dimensionality, chaos and turbulence, multiscale and multiphysics interactions, partial observability, discontinuities, non-Gaussian statistics, intermittency, and extreme events. To address these intrinsic complexities, this thesis integrates heterogeneous sources of information, such as data, physical laws, and domain knowledge, to develop novel SciML and DA methods for solving the associated forward and inverse problems, including spatiotemporal modeling, system identification, data assimilation, and uncertainty quantification.

The contributions of this thesis can be summarized as follows: (1) Developing a continuous modeling framework, termed the neural dynamical operator, for learning complex spatiotemporal dynamical systems such as nonlinear PDEs. The model is resolution-invariant (or discretization-invariant) in both the spatial and temporal domains. A hybrid optimization scheme that combines gradient-based and derivative-free methods is proposed to train the model for both short-term state forecasting and long-term statistical matching. (2) Proposing an online system identification method, termed Causation Entropy Boosting (CEBoosting), to detect changing points (tipping points) and calibrate the model for dynamical systems with regime switching, where the dynamical function or its parameters change abruptly,

based on sequential streaming time series data. (3) Designing a stochastic surrogate modeling framework, constituted by a novel neural network architecture termed the conditional Gaussian Koopman network (CGKN), for modeling partially observed nonlinear dynamical systems to enable efficient nonlinear data assimilation (state estimation) and state forecast with uncertainty quantification. In addition, extensive numerical examples are provided to demonstrate the robustness and efficiency of these computational methods.

The methods developed in this thesis also offer promising directions for further extensions and applications. In Chap. 2, the neural dynamical operator can be employed to model the unresolved components in spatiotemporal closure modeling, complementing the resolved components such as physics-based terms. Additionally, the spatiotemporal resolution-invariance property enables model training and inference on data with irregular sampling in both space and time, e.g., data collected from spatially moving sensors with varying temporal sampling rates. Furthermore, instead of the heuristic alternative hybrid optimization schemes, utilizing ensemble Kalman inversion (or other memory-efficient and chaos-robust optimization methods) to minimize short-term and long-term additive losses may further improve model performance. In Chap. 3, the physical constraints can be naturally added to the CEBoosting algorithm, such as the requirement of the total energy in the quadratic nonlinear terms to be conserved. This energy conservation of the modeled system guarantees the long-term stability of the identified system. Other future work includes the uncertainty quantification of the aggregated causation entropy and the further study of other causality metrics. In Chap. 4, the proposed CGKN demonstrates the potential for real-time field reconstruction of three-dimensional turbulent systems from sparse observations. In addition, its lightweight design and online learning capability provide a viable solution for edge intelligence, where large-scale models are often impractical due to limited storage and computational resources. This framework can be further extended to non-Markovian partially observed systems by incorporating memory, and can be adapted to Lagrangian data streams. Furthermore, potential methodological improvements include more advanced uncertainty quantification methods for DA results and the learning of informative embeddings via more powerful data compression techniques.

Ultimately, this thesis develops systematic computational methods and algorithms on the basis of SciML and DA to enable the modeling, simulation, and analysis of complex dynamical systems across science and engineering. These methods and algorithms contribute to addressing previously intractable and emerging challenges in forward and inverse problems,

demonstrating strong potential for real-world applications.

Appendix A

Algorithm for the Neural Dynamical Operator with Hybrid Optimization

The detailed algorithm of the hybrid optimization is presented in Algorithm 3. In this thesis, we apply this algorithm to the example of Kuramoto—Sivashinsky equation in Section 2.6.3, for an efficient and robust training of neural dynamical operator with both short-term and long-term data.

Algorithm 3 Training neural dynamical operator with the hybrid optimization scheme

Input: $\{\mathbf{u}(t_n)\}_{n=0}^{N}$ and $\tilde{\mathcal{G}}(\cdot;\boldsymbol{\theta})$ ▷ Training data and initial model
Set $(N_{\text{Hybrid}}, N_{\text{SGD}}, N_{\text{EKI}}, N_s, N_l, \lambda, J, c)$
$\boldsymbol{\Theta} \leftarrow$ Empty list
for $n = 1, 2, \ldots, N_{\text{Hybrid}}$ **do**
    for $n_1 = 1, 2, \ldots, N_{\text{SGD}}$ **do** ▷ **SGD training**
        $N_0 \sim \mathbb{U}(0, N - N_s)$
        $\{\tilde{\mathbf{u}}(t_n)\}_{n=N_0}^{N_0+N_s} = \text{ODESolver}(\tilde{\mathcal{G}}(\boldsymbol{\theta}), \mathbf{u}(t_{N_0}), \{t_n\}_{n=N_0}^{N_0+N_s})$
        $L_s = \Sigma_{n=N_0}^{N_0+N_s} \|\mathbf{u}(t_n) - \tilde{\mathbf{u}}(t_n)\|^2$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} L_s$ ▷ Parameters updated by SGD
    end for
    $\{\boldsymbol{\theta}^{(j)}\}_{j=1}^{J} = \boldsymbol{\theta} + \{\boldsymbol{\epsilon}^{(j)}\}_{j=1}^{J}, \quad \boldsymbol{\epsilon}^{(j)} \sim \mathcal{N}(0, 0.1^2 \mathbf{I}_{d_{\boldsymbol{\theta}}})$ ▷ Initialize ensemble parameters
    for $n_2 = 1, 2, \ldots, N_{\text{EKI}}$ **do** ▷ **EKI training**
        $N_0 \sim \mathbb{U}(0, N - N_l)$
        $\mathbf{y} = \beta(\{\tilde{\mathbf{u}}(t_n)\}_{n=N_0}^{N_0+N_l})$ ▷ Long-term statistics
        $\boldsymbol{\Sigma}_{\boldsymbol{\eta}} = c^2 \mathbf{I}_{d_y}$
        $\left\{[\{\tilde{\mathbf{u}}(t_n)\}_{n=0}^{N_l}]^{(j)}\right\}_{j=1}^{J} = \left\{\text{ODESolver}(\tilde{\mathcal{G}}(\boldsymbol{\theta}^{(j)}), \mathbf{u}(t_{N_0}), \{t_n\}_{n=N_0}^{N_0+N_l})\right\}_{j=1}^{J}$
        $\{\mathbf{g}^{(j)}\}_{j=1}^{J} = \left\{\beta([\{\tilde{\mathbf{u}}(t_n)\}_{n=0}^{N_l}]^{(j)})\right\}_{j=1}^{J}$
        $\{\mathbf{y}^{(j)}\}_{j=1}^{J} = \mathbf{y} + \boldsymbol{\eta}^{(j)}, \quad \boldsymbol{\eta}^{(j)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\eta}})$
        $\{\boldsymbol{\theta}^{(j)}\}_{j=1}^{J} \leftarrow \left\{\boldsymbol{\theta}^{(j)} + \boldsymbol{\Sigma}^{\boldsymbol{\theta g}}(\boldsymbol{\Sigma}^{\mathbf{gg}} + \boldsymbol{\Sigma}_{\boldsymbol{\eta}})^{-1}(\mathbf{y}^{(j)} - \mathbf{g}^{(j)})\right\}_{j=1}^{J}$ ▷ EKI updating formula
    end for
    $\boldsymbol{\theta} = \frac{1}{J}\Sigma_{j=1}^{J}\boldsymbol{\theta}^{(j)}$ ▷ Parameters updated by EKI
    Append $\boldsymbol{\theta}$ to $\boldsymbol{\Theta}$
end for
return robust $\boldsymbol{\theta}^{\star}$ from $\boldsymbol{\Theta}$ based on the error history during EKI updating

Appendix B

Algorithm for the Causation Entropy Boosting

Algorithm 4 presents the detailed procedures of the CEBoosting method, including (i) detecting regime switching, (ii) aggregating causation entropy matrix (CEM) until a consistent pattern of CEM is obtained, (iii) identifying a sparse model structure according to the aggregated CEM, and (iv) fitting the model parameters. The consistent pattern of CEM is determined by (3.11), i.e., the aggregated CEM does not change for D data batches. In this work, we choose $\mathsf{D} = 4$ for all the numerical examples.

Algorithm 4 Causation Entropy Boosting

$\mathbf{f} \leftarrow \boldsymbol{\Xi}\boldsymbol{\Phi}$ ▷ Current Model
$D \leftarrow 4$ ▷ Stable Condition of Aggregated CEM
for $k = 1, 2, \ldots$ **do** ▷ Sequential data batches with each of length M
  $\boldsymbol{\Phi}_m \leftarrow \boldsymbol{\Phi}(\mathbf{x}(t_{B_k} + (m-1)\Delta t))$
  $\dot{\mathbf{x}}_m \leftarrow [\mathbf{x}(t_{B_k} + m\Delta t) - \mathbf{x}(t_{B_k} + (m-1)\Delta t)]/\Delta t$
  $\mathbf{r}_m \leftarrow \dot{\mathbf{x}}_m - \boldsymbol{\Xi}\boldsymbol{\Phi}_m, \quad m = 1, \ldots, M-1$ ▷ Estimation of residual dynamics
  $\mathbf{C} \leftarrow C_{\phi_n \to r_i | [\boldsymbol{\Phi} \setminus \phi_n]} \geqslant \overline{C}$
  if $\mathbf{C} = 0$ **then** ▷ Same regime
    $\mathbf{f} \leftarrow \boldsymbol{\Xi}\boldsymbol{\Phi}$ ▷ Keep current model
  else ▷ New regime detected
    $K \leftarrow 0$, $d \leftarrow 1$,
    while $d < D$ **do** ▷ Aggregating CEM until stable for D iterations
      $K \leftarrow K + 1$
      $\mathbf{C} \leftarrow \mathbf{C} + C^{(k+K)}_{\phi_n \to r_i | [\boldsymbol{\Phi} \setminus \phi_n]}$
      $\mathbf{C}^+(K) \leftarrow \frac{1}{K}\mathbf{C} \geqslant \overline{C}$ ▷ Indicator matrix
      if $\mathbf{C}^+(K) = \mathbf{C}^+(K-1)$ **then** ▷ Stable condition
        $d \leftarrow d + 1$
      else ▷ Unstable CEM
        $d \leftarrow 1$
      end if
    end while ▷ Stable CEM
    $K^* \leftarrow K$ ▷ Minimal batches that produce the stable CEM
    $\delta\boldsymbol{\Xi}[\mathbf{C}^+(K^*) = 0] \leftarrow 0$ ▷ Enforcing sparsity by the stable CEM
    $\delta\boldsymbol{\Xi}[\mathbf{C}^+(K^*) = 1] \leftarrow \arg\min_{\delta\boldsymbol{\Xi}} \sum_{m=1}^{MK^*} \|\mathbf{r}_m - \delta\boldsymbol{\Xi}\boldsymbol{\Phi}_m\|^2$
    $\boldsymbol{\Xi} \leftarrow \boldsymbol{\Xi} + \delta\boldsymbol{\Xi}$
    $\mathbf{f}^* \leftarrow \boldsymbol{\Xi}\boldsymbol{\Phi}$
  end if
  return $\mathbf{f}^*$
end for

Appendix C

Algorithm for the Conditional Gaussian Koopman Network

The detailed algorithm for the discrete CGKN is presented in Algorithm 3. The same process can also be applied to continuous CGKN and CGNSDE, with the corresponding formulations.

Algorithm 5 Learning Discrete Conditional Gaussian Koopman Network

Input: $\{\mathbf{u}^{n\star} = (\mathbf{u}_1^{n\star}, \mathbf{u}_2^{n\star})\}_{n=0}^{N}$ and $(\boldsymbol{\varphi}, \boldsymbol{\psi}, \mathbf{F}_1, \mathbf{G}_1, \mathbf{F}_2, \mathbf{G}_2)$ ▷ Data and initial Model

$\boldsymbol{\theta} = \{\boldsymbol{\theta}_{\boldsymbol{\varphi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}, \boldsymbol{\theta}_{\mathbf{F}_1}, \boldsymbol{\theta}_{\mathbf{G}_1}, \boldsymbol{\theta}_{\mathbf{F}_2}, \boldsymbol{\theta}_{\mathbf{G}_2}\}$ ▷ Model parameters

Stage 1:

for ep = 1, 2, ... epochs **do**

  $N_0 \sim \mathbb{U}(0, 1, \ldots, N - N_s)$ ▷ Randomly sample a time step

  $\mathbf{u}_1^{N_0} = \mathbf{u}_1^{N_0\star}$

  $\mathbf{v}^{N_0} = \mathbf{v}^{N_0\star} = \boldsymbol{\varphi}(\mathbf{u}_2^{N_0\star}; \boldsymbol{\theta}_{\boldsymbol{\varphi}})$ ▷ Encode unobserved state to latent state

  for $n = N_0, N_0 + 1, \ldots, N_0 + N_s - 1$ **do** ▷ State forecast for horizon N_s

    $\mathbf{u}_1^{n+1} = \mathbf{F}_1(\mathbf{u}_1^n; \boldsymbol{\theta}_{\mathbf{F}_1}) + \mathbf{G}_1(\mathbf{u}_1^n; \boldsymbol{\theta}_{\mathbf{G}_1})\mathbf{v}^n$

    $\mathbf{v}^{n+1} = \mathbf{F}_2(\mathbf{u}_1^n; \boldsymbol{\theta}_{\mathbf{F}_2}) + \mathbf{G}_2(\mathbf{u}_1^n; \boldsymbol{\theta}_{\mathbf{G}_2})\mathbf{v}^n$

    $\mathbf{u}_2^{n+1} = \boldsymbol{\psi}(\mathbf{v}^{n+1}; \boldsymbol{\theta}_{\boldsymbol{\psi}})$ ▷ Decode latent state to unobserved state

  end for

  $L_{AE} = \frac{1}{N_s+1} \sum_{n=N_0}^{N_0+N_s} \|\mathbf{u}_2^{n\star} - \boldsymbol{\psi}\big(\boldsymbol{\varphi}(\mathbf{u}_2^{\star})\big)\|^2$

  $L_{\mathbf{u}} = \frac{1}{N_s} \sum_{n=N_0+1}^{N_0+N_s} \|\mathbf{u}^{n\star} - \mathbf{u}^n\|^2$ ▷ $\mathbf{u}^n = (\mathbf{u}_1^n, \mathbf{u}_2^n)$

  $L_{\mathbf{v}} = \frac{1}{N_s} \sum_{n=N_0+1}^{N_0+N_s} \|\mathbf{v}^{n\star} - \mathbf{v}^n\|^2$

  $L = \lambda_{AE} L_{AE} + \lambda_{\mathbf{u}} L_{\mathbf{u}} + \lambda_{\mathbf{v}} L_{\mathbf{v}}$

  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} L$ ▷ Gradient descent without DA loss

end for

$\boldsymbol{\sigma}_1 = \mathrm{diag}\Big(\sqrt{\frac{1}{N} \sum_{n=1}^{N} (\mathbf{u}_1^{n\star} - \mathbf{u}_1^n \odot (\mathbf{u}_1^{n\star} - \mathbf{u}_1^n)}\Big)$ ▷ UQ for state forecast via RMSE

$\boldsymbol{\sigma}_2 = \mathrm{diag}\Big(\sqrt{\frac{1}{N} \sum_{n=1}^{N} (\mathbf{v}^{n\star} - \mathbf{v}^n) \odot (\mathbf{v}^{n\star} - \mathbf{v}^n)}\Big)$

$\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$

Stage 2:

for ep = 1, 2, ... epochs2 **do**

  Line 4 - 13

  $N_0' \sim \mathbb{U}(0, 1, \ldots N - N_l)$ ▷ Randomly sample a time step

  for $n = N_0', N_0' + 1, \ldots, N_0' + N_l - 1$ **do** ▷ Data assimilation for horizon N_l

    $\boldsymbol{\mu}_{\mathbf{v}}^{n+1} = \mathbf{F}_2 + \mathbf{G}_2 \boldsymbol{\mu}_{\mathbf{v}}^n + \mathbf{K}^n \big(\mathbf{u}_1^{n+1} - \mathbf{F}_1 - \mathbf{G}_1 \boldsymbol{\mu}_{\mathbf{v}}^n\big)$ ▷ The input of $\mathbf{F}$, $\mathbf{G}$ are ommitted

    $\boldsymbol{\Sigma}_{\mathbf{v}}^{n+1} = \mathbf{G}_2 \boldsymbol{\Sigma}_{\mathbf{v}}^n \mathbf{G}_2^\intercal + \boldsymbol{\sigma}_2 \boldsymbol{\sigma}_2^\intercal - \mathbf{K}^n \mathbf{G}_1 \boldsymbol{\Sigma}_{\mathbf{v}}^n \mathbf{G}_2^\intercal$

  end for

  $L_{DA} = \frac{1}{N_l - N_b} \sum_{n=N'0+N_b}^{N'0+N_l} \|\mathbf{u}_2^{n\star} - \boldsymbol{\mu}^n\|^2$ ▷ N_b is the DA warm-up time

  $L = \lambda_{AE} L_{AE} + \lambda_{\mathbf{u}} L_{\mathbf{u}} + \lambda_{\mathbf{v}} L_{\mathbf{v}} + L_{DA}$

  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} L$ ▷ Gradient descent with DA loss

end for

$\mathbf{r}^n = |\mathbf{u}_2^{n\star} - \boldsymbol{\mu}^n|$ **for** $n = 1, 2, \ldots, N$ ▷ Residual

$\boldsymbol{\theta}_{UQ}^* = \arg\min \frac{1}{N} \sum_{n=1}^{N} \big(\mathbf{r}^n - \mathrm{NN}(\mathbf{u}_1^{n\star}; \boldsymbol{\theta}_{UQ})\big)^2$ ▷ UQ for DA via Residual regression

Output: $\boldsymbol{\theta}^*$, $\boldsymbol{\sigma}$, $\boldsymbol{\theta}_{UQ}^*$ ▷ Trained and estimated parameters

Appendix D

Data Assimilation for Conditional Gaussian Nonlinear System

The general form of dynamical systems from fluid mechanics and geophysics could be represented as:

$$\frac{d\mathbf{u}}{dt} = (L+D)\mathbf{u} + B(\mathbf{u},\mathbf{u}) + \mathbf{F}(t) + \boldsymbol{\sigma}(\mathbf{u},t)\dot{\mathbf{W}}(t) \tag{D.1}$$

with $\mathbf{u} \in \mathbb{C}^N$ is the state variable, $(L+D)\mathbf{u}$ is linear dispersion and dissipation effects, $B(\mathbf{u},\mathbf{u})$ in energy-conserving quadratic form introducing nonlinear effect, $\mathbf{F}(t)$ is a deterministic external forcing term, $\boldsymbol{\sigma} \in \mathbb{C}^{N\times K}$ is noise matrix, and $\dot{\mathbf{W}} \in \mathbb{C}^K$ is white noise.

This general form (D.1) could be approximated by conditional Gaussian non-linear system (CGNS):

$$\begin{aligned} \frac{d\mathbf{X}}{dt} &= \mathbf{a}_0(\mathbf{X},t) + \mathbf{a}_1(\mathbf{X},t)\mathbf{Y}(t) + \mathbf{b}_1(\mathbf{X},t)\dot{\mathbf{W}}_1(t) \\ \frac{d\mathbf{Y}}{dt} &= \mathbf{a}_0(\mathbf{X},t) + \mathbf{a}_1(\mathbf{X},t)\mathbf{Y}(t) + \mathbf{b}_2(\mathbf{X},t)\dot{\mathbf{W}}_2(t) \end{aligned} \tag{D.2}$$

with original state $\mathbf{u}$ be decomposed into $\mathbf{X} \in \mathbb{C}^{N_1}$ and $\mathbf{Y} \in \mathbb{C}^{N_2}$. $\mathbf{a}_0$, $\mathbf{a}_0$, $\mathbf{a}_1$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are vectors or matrix which could depend non-linearly on $\mathbf{X}$ and t. $\dot{\mathbf{W}}_1(t)$ and $\dot{\mathbf{W}}_2(t)$ are independent white noise.

For CGNS (D.2), the optimal filter solution of $\mathbf{Y}(t)$ given $\mathbf{X}(s), s \in [0,T]$ is :

$$\begin{aligned} p(\mathbf{Y}(t)|\mathbf{X}(s), s \leqslant t) &\sim \mathcal{N}(\boldsymbol{\mu}_f(t), \mathbf{R}_f(t)) \\ \frac{d\boldsymbol{\mu}_f}{dt} &= (\mathbf{a}_0 + \mathbf{a}_1\boldsymbol{\mu}_f) + (\mathbf{R}_f\mathbf{a}_1^*)(\mathbf{b}_1\mathbf{b}_1^*)^{-1}(\frac{d\mathbf{X}}{dt} - (\mathbf{a}_0 + \mathbf{a}_1\boldsymbol{\mu}_f)) \\ \frac{d\mathbf{R}_f}{dt} &= \mathbf{a}_1\mathbf{R}_f + \mathbf{R}_f\mathbf{a}_1^* + \mathbf{b}_2\mathbf{b}_2^* - (\mathbf{R}_f\mathbf{a}_1^*)(\mathbf{b}_1\mathbf{b}_1^*)^{-1}(\mathbf{a}_1\mathbf{R}_f) \end{aligned} \tag{D.3}$$

(The notation $\cdot^*$ is the complex conjugate transpose)

The above filter solution only exploits information up to current time, not the entire time interval. A smoother solution which utilize whole information in time interval would be more accurate:

$$\begin{aligned} p(\mathbf{Y}(t)|\mathbf{X}(s), s \in [0,T]) &\sim \mathcal{N}(\boldsymbol{\mu}_s(t), \mathbf{R}_s(t)) \\ \frac{\overleftarrow{d\boldsymbol{\mu}_s}}{dt} &= -\mathbf{a}_0 - \mathbf{a}_1\boldsymbol{\mu}_s + (\mathbf{b}_2\mathbf{b}_2^*)\mathbf{R}_f^{-1}(\boldsymbol{\mu}_f - \boldsymbol{\mu}_s) \\ \frac{\overleftarrow{d\mathbf{R}_s}}{dt} &= -(\mathbf{a}_1 + (\mathbf{b}_2\mathbf{b}_2^*)\mathbf{R}_f^{-1})\mathbf{R}_s - \mathbf{R}_s(\mathbf{a}_1^* + (\mathbf{b}_2\mathbf{b}_2^*)\mathbf{R}_f) + \mathbf{b}_2\mathbf{b}_2^* \end{aligned} \tag{D.4}$$

(The notation $\overleftarrow{d\cdot}/dt$ means solving (D.4) backward from endpoint of interval [0,T] with $\boldsymbol{\mu}_s(T) = \boldsymbol{\mu}_f(T)$ and $\mathbf{R}_s(T) = \mathbf{R}_f(T)$)

The conditional sampling formula of $\mathbf{Y}(t)$ given $\mathbf{X}(s), s \in [0,T]$ could then be derived:

$$\frac{\overleftarrow{d\mathbf{Y}}}{dt} = \frac{\overleftarrow{d\boldsymbol{\mu}_s}}{dt} - (\mathbf{a}_1 + (\mathbf{b}_2\mathbf{b}_2^*)\mathbf{R}_f^{-1})(\mathbf{Y} - \boldsymbol{\mu}_s) + \mathbf{b}_2\dot{\mathbf{W}}_{\mathbf{Y}}(t) \tag{D.5}$$

The data assimilation has been used in Section 3.6.4 (SPEKF model) in Chapter 3. The SPEKF model (3.17) is following the structure of conditional Gaussian non-linear system (D.2): $\mathbf{X}$ corresponds to u and $\mathbf{Y}$ corresponds to $\begin{bmatrix}\gamma & \omega & b\end{bmatrix}^T$ with $\mathbf{a}_0 = -\hat{\gamma}u + i\hat{\omega}u + \hat{b}$, $\mathbf{a}_1 = \begin{bmatrix}-u & iu & 1\end{bmatrix}$, $\mathbf{b}_1 = \sigma_u$, $\dot{\mathbf{W}}_1 = \dot{W}_u$, $\mathbf{a}_0 = \mathbf{0}$, $\mathbf{a}_1 = \begin{bmatrix}-d_\gamma & 0 & 0\\ 0 & -d_\omega & 0\\ 0 & 0 & -d_b\end{bmatrix}$, $\mathbf{b}_2 = \begin{bmatrix}\sigma_\gamma & 0 & 0\\ 0 & \sigma_\omega & 0\\ 0 & 0 & \sigma_b\end{bmatrix}$, $\dot{\mathbf{W}}_2 = \begin{bmatrix}\dot{W}_\gamma & \dot{W}_\omega & \dot{W}_b\end{bmatrix}^T$. With the observed variable u, three hidden variables γ, ω, b will be sampled according to the sampling formula (D.5), and presented in Fig 3.8. The smoother estimation in Fig 3.9 is from the smoother formula (D.4). The sampled trajectory is the

one that satisfies the posterior distribution at each time instant and considers the temporal correlation at different time instants. The formula is used in generating the trajectories of the unobserved variables for computing the causation entropy in Section 3.6.4.

Appendix E

A Quick Summary of the ENSO SPDE Test Model

The physics-based reference ENSO model [368] discussed in this paper is built based on the dynamics of atmosphere-ocean interactions in the tropical Pacific. The model consists of two main components: a deterministic dynamical core that captures the essential physics of ENSO and stochastic parameterizations that trigger the extreme events showing the atmospheric and oceanic processes occurring at different timescales.

The dynamical core describes the interactions between the atmosphere, the ocean, and the SST through three key mechanisms. First, the atmosphere drives ocean circulation through wind stress, where the wind busts generate currents and affect the thermocline depth. Second, the ocean affects the atmosphere through latent heat, which is proportional to SST. Third, the ocean modulates SST through changes in the thermocline depth and zonal currents.

The atmospheric component is based on a non-dissipative Matsuno-Gill type atmosphere model [388, 389]. This model is a simplified representation of atmospheric dynamics in the tropics, designed to capture large-scale motions. A key feature in the model of (E.1) is the latent heating E_q, which is proportional to the sea surface temperature and drives the atmospheric circulation. For the ocean component, the model employs a simple shallow-water system [390]. The model focuses on the dynamics of an active upper layer. The upper layer dynamics are driven by surface wind stress from the atmosphere. This wind stress causes currents and waves, representing momentum transfer from the atmosphere to the

ocean. These assumptions result in the reduced shallow-water equations shown in (E.2). The SST equation is derived from an SST budget equation [391]. These include vertical temperature changes due to thermocline variations, horizontal heat transport by ocean currents, and thermal damping through air-sea heat exchange. These essential mechanisms are captured in (E.3).

The dynamical core is a coupled atmosphere-ocean-SST system:

Atmosphere:

$$\begin{aligned} -yv - \partial_x \theta &= 0 \\ yu - \partial_y \theta &= 0 \\ -(\partial_x u + \partial_y v) &= E_q/(1-\bar{Q}) \end{aligned} \tag{E.1}$$

Ocean:

$$\begin{aligned} \partial_t U - c_1 YV + c_1 \partial_x H &= c_1 \tau_x \\ YU + \partial_Y H &= 0 \\ \partial_t H + c_1 (\partial_x U + \partial_Y V) &= 0 \end{aligned} \tag{E.2}$$

SST:

$$\partial_t T = -c_1 \zeta E_q + c_1 \eta_1 H + c_1 I \eta_2 U. \tag{E.3}$$

In the atmosphere component (E.1), u, v, and θ represent the zonal wind speeds, meridional wind speeds, and the potential temperature, respectively. In the ocean components (E.2), U, V are zonal and meridional ocean currents, H is the thermocline depth. Besides, T represents the SST. These variables are all anomalies. The t represents the interannual time coordinate, x is the zonal coordinate, and y and Y are the meridional components for the atmosphere and ocean components. The equations include various parameters such as latent heat $E_q = \alpha_q T$, wind stress τ_x, latent heating exchange coefficient ζ, and the strengths of the thermocline η_1 and zonal advective feedback η_2. The relative magnitudes of η_1 and η_2 differ between the Eastern Pacific (EP) and Central Pacific (CP) due to variations in thermocline depth and zonal SST gradients. The model assumes periodic boundary conditions for the atmosphere and reflection boundary conditions for the Pacific Ocean. Since ENSO is primarily a phenomenon near the equator, we further project the model to the leading basis in the meridional direction.

While this deterministic dynamical core captures the basic ENSO physics, additional processes at different timescales play crucial roles in generating realistic features of ENSO diversity. These additional effects can be effectively characterized by stochastic parameterizations, which describe the intraseasonal wind variations and the decadal changes in the background conditions.

The wind stress τ_x consists of two components, the atmospheric circulation u and the stochastic wind bursts u_p, which is localized in the western Pacific (WP) and has the following structure:

$$u_p(x,y,t) = a_p(t)s_p(x)\phi_0(y) \tag{E.4}$$

where $\phi_0(y)$ is the leading meridional basis, $s_p(x)$ is a fixed spatial structure, and $a_p(t)$ is the wind burst amplitude governed by a stochastic process:

$$\frac{da_p}{dt} = -d_p a_p + \sigma_p(T_C)\dot{W}_p \tag{E.5}$$

Here, d_p is the damping term, $\dot{W}_p$ is a white noise source, and $\sigma_p(T_C)$ is the state-dependent noise strength, which is a function of the SST averaged over the CP. In the absence of seasonal cycle and decadal influence, the noise strength is parameterized as $\sigma_p(T_C) = 1.6(\tanh(T_C)+1)$. This state-dependent noise coefficient implies that wind burst activity is more active during El Niño events due to the eastward extension of the warm pool. The intraseasonal wind bursts not only affect the interannual variability but are also modulated by it.

Finally, the decadal variability is driven by a simple stochastic process:

$$\frac{dI}{dt} = -\lambda(I-m) + \sigma_I(I)\dot{W}_I, \tag{E.6}$$

where λ is the damping term set to 5 years^{-1}, representing the decadal time scale. $\sigma_I(I)$ is the state-dependent noise strength, and $\dot{W}_I$ is the white noise source. The state-dependent noise coefficient allows the distribution of I to be non-Gaussian, with I representing the strength of the easterly trade wind in the lower level of the Walker circulation on the decadal time scale. A uniform distribution between $[0,1]$ is adopted for I, with larger values corresponding to stronger easterly trade winds.

Bibliography

[1] Jürgen Jost. *Dynamical systems: examples of complex behaviour*. Springer Science & Business Media, 2005.

[2] Stephen Wiggins. *Introduction to applied nonlinear dynamical systems and chaos*, volume 2. Springer Science & Business Media, 2003.

[3] Andrew Majda and Xiaoming Wang. *Nonlinear dynamics and statistical theories for basic geophysical flows*. Cambridge University Press, 2006.

[4] Nan Chen. *Stochastic Methods for Modeling and Predicting Complex Dynamical Systems: Uncertainty Quantification, State Estimation, and Reduced-Order Models*. Springer Nature, 2023.

[5] Konstantinos Vogiatzoglou, Costas Papadimitriou, Vasilis Bontozoglou, and Konstantinos Ampountolas. Physics-informed neural networks for parameter learning of wildfire spreading. *Computer Methods in Applied Mechanics and Engineering*, 434:117545, 2025.

[6] Zihao Wang, Guiyong Zhang, Tiezhi Sun, and Bo Zhou. Development and application of a fluid mechanics analysis framework based on complex network theory. *Computer Methods in Applied Mechanics and Engineering*, 435:117677, 2025.

[7] Henk A Dijkstra. *Nonlinear climate dynamics*. Cambridge University Press, 2013.

[8] Kevin E Trenberth, John T Fasullo, and Theodore G Shepherd. Attribution of climate extreme events. *Nature Climate Change*, 5(8):725–730, 2015.

[9] HK Moffatt. Extreme events in turbulent flow. *Journal of Fluid Mechanics*, 914:F1, 2021.

[10] Andrew Majda. *Introduction to PDEs and Waves for the Atmosphere and Ocean*, volume 9. American Mathematical Soc., 2003.

[11] Andrew J Majda and Nan Chen. Model error, information barriers, state estimation and prediction in complex multiscale systems. *Entropy*, 20(9):644, 2018.

[12] Paul Manneville and Yves Pomeau. Intermittency and the Lorenz model. *Physics Letters A*, 75(1-2):1–2, 1979.

[13] Jin-Long Wu, Matthew E Levine, Tapio Schneider, and Andrew Stuart. Learning about structural errors in models of complex dynamical systems. *Journal of Computational Physics*, page 113157, 2024.

[14] Nan Chen and Di Qi. A physics-informed data-driven algorithm for ensemble forecast of complex turbulent systems. *Applied Mathematics and Computation*, 466:128480, 2024.

[15] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

[16] Hayden Schaeffer, Russel Caflisch, Cory D Hauck, and Stanley Osher. Sparse dynamics for partial differential equations. *Proceedings of the National Academy of Sciences*, 110(17):6634–6639, 2013.

[17] Andrew J Majda and Di Qi. Strategies for reduced-order models for predicting the statistical responses and uncertainty quantification in complex turbulent dynamical systems. *SIAM Review*, 60(3):491–549, 2018.

[18] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[19] Bernd R Noack, Marek Morzynski, and Gilead Tadmor. *Reduced-order modelling for flow control*, volume 528. Springer Science & Business Media, 2011.

[20] Nan Chen and Yinling Zhang. A causality-based learning approach for discovering the underlying dynamics of complex systems from partial observations with stochastic parameterization. *Physica D: Nonlinear Phenomena*, 449:133743, 2023.

[21] Chuanqi Chen, Nan Chen, and Jin-Long Wu. CEBoosting: Online sparse identification of dynamical systems with regime switching by causation entropy boosting. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(8), 2023.

[22] Nan Chen and Honghu Liu. Minimum reduced-order models via causal inference. *Nonlinear Dynamics*, pages 1–25, 2024.

[23] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Solving and learning nonlinear PDEs with Gaussian processes. *Journal of Computational Physics*, 447:110668, 2021.

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

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

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

[27] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995.

[28] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, 2, 1989.

[29] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[30] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019.

[31] Rose Yu and Rui Wang. Learning dynamical systems from data: An introduction to physics-guided deep learning. *Proceedings of the National Academy of Sciences*, 121(27):e2311808121, 2024.

[32] Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. *Physical Review Fluids*, 2(3):034603, 2017.

[33] Jin-Long Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 3(7):074602, 2018.

[34] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[35] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[36] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

[39] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[40] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[41] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[42] Chuanqi Chen and Jin-Long Wu. Neural dynamical operator: Continuous spatial-temporal model with gradient-based and derivative-free optimization methods. *Journal of Computational Physics*, 520:113480, 2025.

[43] Zecheng Zhang. MODNO: Multi-operator learning with distributed neural operators. *Computer Methods in Applied Mechanics and Engineering*, 431:117229, 2024.

[44] Guang Lin, Christian Moya, and Zecheng Zhang. B-deeponet: An enhanced Bayesian deeponet for solving noisy parametric pdes using accelerated replica exchange sgld. *Journal of Computational Physics*, 473:111713, 2023.

[45] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

[46] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[47] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[48] Jin-Long Wu, Karthik Kashinath, Adrian Albert, Dragos Chirila, Heng Xiao, et al. Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. *Journal of Computational Physics*, 406:109209, 2020.

[49] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[50] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[51] Xinghao Dong, Chuanqi Chen, and Jin-Long Wu. Data-driven stochastic closure modeling via conditional diffusion model and neural operator. *Journal of Computational Physics*, 534:114005, 2025.

[52] Pan Du, Meet Hemant Parikh, Xiantao Fan, Xin-Yang Liu, and Jian-Xun Wang. Conditional neural field latent diffusion model for generating spatiotemporal turbulence. *Nature Communications*, 15(1):10416, 2024.

[53] Stamatis Stamatelopoulos and Themistoklis P Sapsis. Can diffusion models capture extreme event statistics? *Computer Methods in Applied Mechanics and Engineering*, 435:117589, 2025.

[54] Han Gao, Sebastian Kaltenbach, and Petros Koumoutsakos. Generative learning of the solution of parametric partial differential equations using guided diffusion models and virtual observations. *Computer Methods in Applied Mechanics and Engineering*, 435:117654, 2025.

[55] Burr Settles. Active learning literature survey, 2009.

[56] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Learning active learning from data. *Advances in neural information processing systems*, 30, 2017.

[57] Xun Huan and Youssef M Marzouk. Simulation-based optimal Bayesian experimental design for nonlinear systems. *Journal of Computational Physics*, 232(1):288–317, 2013.

[58] Udo Von Toussaint. Bayesian inference in physics. *Reviews of Modern Physics*, 83(3):943–999, 2011.

[59] Harshit Kapadia, Lihong Feng, and Peter Benner. Active-learning-driven surrogate modeling for efficient simulation of parametric nonlinear systems. *Computer Methods in Applied Mechanics and Engineering*, 419:116657, 2024.

[60] Anthony Atkinson, Alexander Donev, and Randall Tobias. *Optimum experimental designs, with SAS*, volume 34. OUP Oxford, 2007.

[61] Huchen Yang, Chuanqi Chen, and Jin-Long Wu. Active learning of model discrepancy with bayesian experimental design. *Computer Methods in Applied Mechanics and Engineering*, 446:118198, 2025.

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

[63] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. *Journal of basic engineering*, 83(1):95–108, 1961.

[64] Andrew J Majda and John Harlim. *Filtering complex turbulent systems*. Cambridge University Press, 2012.

[65] Kody Law, Andrew Stuart, and Konstantinos Zygalakis. *Data assimilation: a mathematical introduction*, volume 62. Springer, 2015.

[66] Tapio Schneider, Andrew M Stuart, and Jin-Long Wu. Learning stochastic closures using ensemble Kalman inversion. *Transactions of Mathematics and Its Applications*, 5(1):tnab003, 2021.

[67] Tapio Schneider, Andrew M Stuart, and Jin-Long Wu. Ensemble Kalman inversion for sparse learning of dynamical systems from time-averaged data. *Journal of Computational Physics*, 470:111559, 2022.

[68] Marco A Iglesias, Kody JH Law, and Andrew M Stuart. Ensemble Kalman methods for inverse problems. *Inverse Problems*, 29(4):045001, 2013.

[69] Nan Chen, Yuchen Li, and Evelyn Lunasin. An efficient continuous data assimilation algorithm for the sabra shell model of turbulence. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(10), 2021.

[70] Nan Chen and Andrew J Majda. Efficient statistically accurate algorithms for the Fokker–Planck equation in large dimensions. *Journal of Computational Physics*, 354:242–268, 2018.

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

[72] Kay Bergemann and Sebastian Reich. An ensemble Kalman-Bucy filter for continuous data assimilation. *Meteorologische Zeitschrift*, 21(3):213, 2012.

[73] Geir Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003.

[74] Jeffrey S Whitaker and Thomas M Hamill. Ensemble data assimilation without perturbed observations. *Monthly weather review*, 130(7):1913–1924, 2002.

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

[76] Changhong Mou, Birgul Koc, Omer San, Leo G Rebholz, and Traian Iliescu. Data-driven variational multiscale reduced order models. *Computer Methods in Applied Mechanics and Engineering*, 373:113470, 2021.

[77] Benjamin Peherstorfer and Karen Willcox. Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21–41, 2015.

[78] Saddam Hijazi, Giovanni Stabile, Andrea Mola, and Gianluigi Rozza. Data-driven POD-Galerkin reduced order model for turbulent flows. *Journal of Computational Physics*, 416:109513, 2020.

[79] Kevin K Lin and Fei Lu. Data-driven model reduction, Wiener projections, and the Koopman-Mori-Zwanzig formalism. *Journal of Computational Physics*, 424:109864, 2021.

[80] Wei Xiao, Xiaojing Liu, Jianhua Zu, Xiang Chai, Hui He, and Tengfei Zhang. Operator inference driven data assimilation for high fidelity neutron transport. *Computer Methods in Applied Mechanics and Engineering*, 430:117214, 2024.

[81] Changhong Mou, Leslie M Smith, and Nan Chen. Combining stochastic parameterized reduced-order models with machine learning for data assimilation and uncertainty quantification with partial observations. *Journal of Advances in Modeling Earth Systems*, 15(10):e2022MS003597, 2023.

[82] Judith Berner, Ulrich Achatz, Lauriane Batte, Lisa Bengtsson, Alvaro de la Cámara, Hannah M Christensen, Matteo Colangeli, Danielle RB Coleman, Daan Crommelin, Stamen I Dolaptchiev, et al. Stochastic parameterization: Toward a new view of weather and climate models. *Bulletin of the American Meteorological Society*, 98(3):565–588, 2017.

[83] PierGianLuca Porta Mana and Laure Zanna. Toward a stochastic parameterization of ocean mesoscale eddies. *Ocean Modelling*, 79:1–20, 2014.

[84] Andrew Dawson and TN Palmer. Simulating weather regimes: Impact of model resolution and stochastic parameterization. *Climate Dynamics*, 44:2177–2193, 2015.

[85] Nan Chen, Changhong Mou, Leslie M Smith, and Yeyu Zhang. A stochastic precipitating quasi-geostrophic model. *Physics of Fluids*, 36(11), 2024.

[86] Jeffrey L. Anderson. An ensemble adjustment Kalman filter for data assimilation. *Monthly Weather Review*, 129(12):2884–2903, 2001.

[87] Quentin Malartic, Alban Farchi, and Marc Bocquet. State, global, and local parameter estimation using local ensemble Kalman filters: Applications to online machine learning of chaotic dynamics. *Quarterly Journal of the Royal Meteorological Society*, 148(746):2167–2193, 2022.

[88] Sibo Cheng, César Quilodrán-Casas, Said Ouala, Alban Farchi, Che Liu, Pierre Tandeo, Ronan Fablet, Didier Lucor, Bertrand Iooss, Julien Brajard, et al. Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review. *IEEE/CAA Journal of Automatica Sinica*, 10(6):1361–1387, 2023.

[89] Stephan Rasp, Michael S Pritchard, and Pierre Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39):9684–9689, 2018.

[90] Massimo Bonavita and Patrick Laloyaux. Machine learning for model error inference and correction. *Journal of Advances in Modeling Earth Systems*, 12(12):e2020MS002232, 2020.

[91] Amina Benaceur and Barbara Verfürth. Statistical variational data assimilation. *Computer Methods in Applied Mechanics and Engineering*, 432:117402, 2024.

[92] Sibo Cheng, Yilin Zhuang, Lyes Kahouadji, Che Liu, Jianhua Chen, Omar K Matar, and Rossella Arcucci. Multi-domain encoder–decoder neural networks for latent data assimilation in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 430:117201, 2024.

[93] Ashesh Chattopadhyay, Ebrahim Nabizadeh, Eviatar Bach, and Pedram Hassanzadeh. Deep learning-enhanced ensemble-based data assimilation for high-dimensional nonlinear dynamical systems. *Journal of Computational Physics*, 477:111918, 2023.

[94] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided Kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

[95] Pierre Boudier, Anthony Fillion, Serge Gratton, Selime Gürol, and Sixin Zhang. Data assimilation networks. *Journal of Advances in Modeling Earth Systems*, 15(4):e2022MS003353, 2023.

[96] Geir Evensen et al. *Data assimilation: the ensemble Kalman filter*, volume 2. Springer, 2009.

[97] Nikola B Kovachki and Andrew M Stuart. Ensemble Kalman inversion: a derivative-free technique for machine learning tasks. *Inverse Problems*, 35(9):095005, 2019.

[98] Chuanqi Chen, Zhongrui Wang, Nan Chen, and Jin-Long Wu. Modeling partially observed nonlinear dynamical systems and efficient data assimilation via discrete-time conditional Gaussian Koopman network. *Computer Methods in Applied Mechanics and Engineering*, 445:118189, 2025.

[99] Zhongrui Wang, Chuanqi Chen, Jin-Long Wu, and Nan Chen. A Lagrangian conditional Gaussian Koopman network for data assimilation and prediction. *arXiv preprint arXiv:2603.14115*, 2026.

[100] Adrian E Gill. *Atmosphere-ocean dynamics*, volume 30. Academic press, 1982.

[101] Harindra JS Fernando. Turbulent mixing in stratified fluids. *Annual Review of Fluid Mechanics*, 23(1):455–493, 1991.

[102] Geoffrey K Vallis. *Atmospheric and oceanic fluid dynamics*. Cambridge University Press, 2017.

[103] Paul E Dimotakis. Turbulent mixing. *Annu. Rev. Fluid Mech.*, 37:329–356, 2005.

[104] Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9784–9790. IEEE, 2019.

[105] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[106] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[107] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[108] Abhinav Gupta and Pierre FJ Lermusiaux. Neural closure models for dynamical systems. *Proceedings of the Royal Society A*, 477(2252):20201004, 2021.

[109] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[110] Chuanqi Chen, Nan Chen, and Jin-Long Wu. CGNSDE: Conditional Gaussian neural stochastic differential equation for modeling complex systems and data assimilation. *Computer Physics Communications*, 304:109302, 2024.

[111] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[112] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, 1993.

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

[114] 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.

[115] 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.

[116] Nils P Wedi, Inna Polichtchouk, Peter Dueben, Valentine G Anantharaj, Peter Bauer, Souhail Boussetta, Philip Browne, Willem Deconinck, Wayne Gaudin, Ioan Hadade, et al. A baseline for global weather and climate simulations at 1 km resolution. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002192, 2020.

[117] Fernando Porté-Agel, Yu-Ting Wu, Hao Lu, and Robert J Conzemius. Large-eddy simulation of atmospheric boundary layer flow through wind turbines and wind farms. *Journal of Wind Engineering and Industrial Aerodynamics*, 99(4):154–168, 2011.

[118] Hao Lu and Fernando Porté-Agel. Large-eddy simulation of a very large wind farm in a stable atmospheric boundary layer. *Physics of Fluids*, 23(6):065101, 2011.

[119] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[120] Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. Neural operator: Graph kernel network for partial differential equations. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[121] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[122] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[123] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in Neural Information Processing Systems*, 34:24048–24062, 2021.

[124] Dhruv Patel, Deep Ray, Michael RA Abdelmalik, Thomas JR Hughes, and Assad A Oberai. Variationally mimetic operator networks. *arXiv preprint arXiv:2209.12871*, 2022.

[125] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[126] Lianghao Cao, Thomas O'Leary-Roseberry, Prashant K Jha, J Tinsley Oden, and Omar Ghattas. Residual-based error correction for neural operator accelerated infinite-dimensional Bayesian inverse problems. *Journal of Computational Physics*, 486:112104, 2023.

[127] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

[128] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, 2021.

[129] Thomas O'Leary-Roseberry, Peng Chen, Umberto Villa, and Omar Ghattas. Derivate informed neural operator: An efficient framework for high-dimensional parametric derivative learning. *arXiv preprint arXiv:2206.10745*, 2022.

[130] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *arXiv preprint arXiv:2207.05209*, 2022.

[131] Guang Lin, Christian Moya, and Zecheng Zhang. Learning the dynamical response of nonlinear non-autonomous dynamical systems with deep operator neural networks. *Engineering Applications of Artificial Intelligence*, 125:106689, 2023.

[132] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39):e2310142120, 2023.

[133] Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Prose: Predicting operators and symbolic expressions using multimodal transformers. *arXiv preprint arXiv:2309.16816*, 2023.

[134] Woojin Cho, Seunghyeon Cho, Hyundong Jin, Jinsung Jeon, Kookjin Lee, Sanghyun Hong, Dongeun Lee, Jonghyun Choi, and Noseong Park. Operator-learning-inspired modeling of neural ordinary differential equations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10):11543–11551, 2024.

[135] W. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[136] Romit Maulik, Arvind Mohan, Bethany Lusch, Sandeep Madireddy, Prasanna Balaprakash, and Daniel Livescu. Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D: Nonlinear Phenomena*, 405:132368, 2020.

[137] Gavin D Portwood, Peetak P Mitra, Mateus Dias Ribeiro, Tan Minh Nguyen, Balasubramanya T Nadiga, Juan A Saenz, Michael Chertkov, Animesh Garg, Anima Anandkumar, Andreas Dengel, et al. Turbulence forecasting via neural ODE. *arXiv preprint arXiv:1911.05180*, 2019.

[138] Alec J Linot and Michael D Graham. Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(7):073110, 2022.

[139] Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics and Control Conference*, pages 263–277. PMLR, 2022.

[140] Alec J Linot, Joshua W Burby, Qi Tang, Prasanna Balaprakash, Michael D Graham, and Romit Maulik. Stabilized neural ordinary differential equations for long-time forecasting of dynamical systems. *Journal of Computational Physics*, 474:111838, 2023.

[141] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

[142] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[143] Charles C Margossian. A review of automatic differentiation and its efficient implementation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1305, 2019.

[144] Qiqi Wang, Rui Hu, and Patrick Blonigan. Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations. *Journal of Computational Physics*, 267:210–224, 2014.

[145] Jin-Long Wu, Matthew E Levine, Tapio Schneider, and Andrew Stuart. Learning about structural errors in models of complex dynamical systems. *arXiv preprint arXiv:2401.00035*, 2023.

[146] Claudia Schillings and Andrew M Stuart. Analysis of the ensemble Kalman filter for inverse problems. *SIAM Journal on Numerical Analysis*, 55(3):1264–1290, 2017.

[147] Zhiyan Ding and Qin Li. Ensemble Kalman inversion: mean-field limit and convergence analysis. *Statistics and Computing*, 31:1–21, 2021.

[148] Edoardo Calvello, Sebastian Reich, and Andrew M Stuart. Ensemble Kalman methods: A mean field perspective. *arXiv preprint arXiv:2209.11371*, 2022.

[149] David J Albers, Paul-Adrien Blancquart, Matthew E Levine, Elnaz Esmaeilzadeh Seylabi, and Andrew Stuart. Ensemble Kalman methods with constraints. *Inverse Problems*, 35(9):095007, 2019.

[150] Neil K Chada, Andrew M Stuart, and Xin T Tong. Tikhonov regularization within ensemble Kalman inversion. *SIAM Journal on Numerical Analysis*, 58(2):1263–1294, 2020.

[151] Yoonsang Lee. l_p regularization for ensemble Kalman inversion. *SIAM Journal on Scientific Computing*, 43(5):A3417–A3437, 2021.

[152] Xin-Lei Zhang, Carlos Michelén-Ströfer, and Heng Xiao. Regularized ensemble Kalman methods for inverse problems. *Journal of Computational Physics*, 416:109517, 2020.

[153] Alfredo Garbuno-Inigo, Franca Hoffmann, Wuchen Li, and Andrew M Stuart. Interacting Langevin diffusions: Gradient structure and ensemble Kalman sampler. *SIAM Journal on Applied Dynamical Systems*, 19(1):412–441, 2020.

[154] Daniel Zhengyu Huang, Tapio Schneider, and Andrew M Stuart. Iterated Kalman methodology for inverse problems. *Journal of Computational Physics*, 463:111262, 2022.

[155] Lucas Böttcher. Gradient-free training of neural ODEs for system identification and control using ensemble Kalman inversion. *arXiv preprint arXiv:2307.07882*, 2023.

[156] Johannes Martinus Burgers. A mathematical model illustrating the theory of turbulence. *Advances in Applied Mechanics*, 1:171–199, 1948.

[157] Eberhard Hopf. The partial differential equation $u_t + uu_x = \mu_{xx}$. *Communications on Pure and Applied mathematics*, 3(3):201–230, 1950.

[158] Jérémie Bec and Konstantin Khanin. Burgers turbulence. *Physics Reports*, 447(1-2):1–66, 2007.

[159] Mehran Kardar, Giorgio Parisi, and Yi-Cheng Zhang. Dynamic scaling of growing interfaces. *Physical Review Letters*, 56(9):889, 1986.

[160] E Weinan, Konstantin Khanin, Alexander Mazel, and Ya Sinai. Invariant measures for Burgers equation with stochastic forcing. *Annals of Mathematics*, pages 877–960, 2000.

[161] Dirk Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73(4):1067, 2001.

[162] Takashi Nagatani. The physics of traffic jams. *Reports on Progress in Physics*, 65(9):1331, 2002.

[163] Claude Navier. *Mémoire sur les lois du mouvement des fluides.* éditeur inconnu, 1822.

[164] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[165] David J Acheson. Elementary fluid dynamics, 1991.

[166] Roger Temam. *Navier-Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Soc., 2001.

[167] 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.

[168] Gi Siv Ashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—I. Derivation of basic equations. In *Dynamics of Curved Fronts*, pages 459–488. Elsevier, 1988.

[169] 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.

[170] Ioannis G Kevrekidis, Basil Nicolaenko, and James C Scovel. Back in the saddle again: a computer assisted study of the Kuramoto–Sivashinsky equation. *SIAM Journal on Applied Mathematics*, 50(3):760–790, 1990.

[171] Qing Wang, Sanjeev R Kulkarni, and Sergio Verdú. Divergence estimation for multidimensional densities via k-Nearest-Neighbor distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405, 2009.

[172] Andrew J Majda. *Introduction to turbulent dynamical systems in complex systems*. Springer, 2016.

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

[174] Dumitru Baleanu, José António Tenreiro Machado, and Albert CJ Luo. *Fractional dynamics and control*. Springer Science & Business Media, 2011.

[175] Thomas Deisboeck and J Yasha Kresh. *Complex systems science in biomedicine*. Springer Science & Business Media, 2007.

[176] J Stelling, A Kremling, M Ginkel, K Bettenbrock, and ED Gilles. *of Book: Foundations of systems biology*. MIT press, 2001.

[177] Sarah A Sheard and Ali Mostashari. Principles of complex systems for systems engineering. *Systems Engineering*, 12(4):295–311, 2009.

[178] Jason M Amundson, Mark Fahnestock, Martin Truffer, Jed Brown, Martin P Lüthi, and Roman J Motyka. Ice mélange dynamics and implications for terminus stability, Jakobshavn Isbræ, greenland. *Journal of Geophysical Research: Earth Surface*, 115(F1), 2010.

[179] Andrew J Majda and Boris Gershgorin. Elementary models for turbulent diffusion with complex physical features: eddy diffusivity, spectrum and intermittency. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1982):20120184, 2013.

[180] Arunn V Holden, Mario Markus, and Hans G Othmer. *Nonlinear wave processes in excitable media*, volume 244. Springer, 2013.

[181] Benjamin Lindner, Jordi Garcıa-Ojalvo, Alexander Neiman, and Lutz Schimansky-Geier. Effects of noise in excitable systems. *Physics reports*, 392(6):321–424, 2004.

[182] Nan Chen, Andrew J Majda, and Xin T Tong. Spatial localization for nonlinear dynamical stochastic models for excitable media. *Chinese Annals of Mathematics, Series B*, 40(6):891–924, 2019.

[183] William K-M Lau and Duane E Waliser. *Intraseasonal variability in the atmosphere-ocean climate system*. Springer Science & Business Media, 2011.

[184] Allan J Clarke. *An introduction to the dynamics of El Niño and the Southern Oscillation*. Elsevier, 2008.

[185] Eduardo G Altmann and Holger Kantz. Recurrence time analysis, long-term correlations, and extreme events. *Physical Review E*, 71(5):056106, 2005.

[186] CA Bronkhorst, H Cho, PW Marcy, SA Vander Wiel, S Gupta, D Versino, V Anghel, and GT Gray III. Local micro-mechanical stress conditions leading to pore nucleation during dynamic loading. *International Journal of Plasticity*, 137:102903, 2021.

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

[188] Michal Branicki, Nan Chen, and Andrew J Majda. Non-Gaussian test models for prediction and state estimation with model errors. *Chinese Annals of Mathematics, Series B*, 34(1):29–64, 2013.

[189] David A Freedman. *Statistical models: theory and practice*. Cambridge University Press, 2009.

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

[191] Andrew J Majda and John Harlim. Physics constrained nonlinear regression models for time series. *Nonlinearity*, 26(1):201, 2012.

[192] John Harlim, Adam Mahdi, and Andrew J Majda. An ensemble Kalman filter for statistical estimation of physics constrained nonlinear regression models. *Journal of Computational Physics*, 257:782–812, 2014.

[193] Nan Chen and Andrew J Majda. Conditional Gaussian systems for multiscale nonlinear stochastic systems: Prediction, state estimation and uncertainty quantification. *Entropy*, 20(7):509, 2018.

[194] Nan Chen, Yingda Li, and Honghu Liu. Conditional Gaussian nonlinear system: A fast preconditioner and a cheap surrogate model for complex nonlinear systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(5):053122, 2022.

[195] Shady E Ahmed, Suraj Pawar, Omer San, Adil Rasheed, Traian Iliescu, and Bernd R Noack. On closures for reduced order models – A spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, 33(9):091301, 2021.

[196] Urban Fasel, J Nathan Kutz, Bingni W Brunton, and Steven L Brunton. Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A*, 478(2260):20210904, 2022.

[197] Stephen A Billings and Hua-Liang Wei. Sparse model identification using a forward orthogonal regression algorithm aided by mutual information. *IEEE Transactions on Neural Networks*, 18(1):306–310, 2007.

[198] Rambod Mojgani, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Discovery of interpretable structural model errors by combining bayesian sparse regression and data assimilation: A chaotic Kuramoto–Sivashinsky test case. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(6):061105, 2022.

[199] Markus Quade, Markus Abel, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for rapid model recovery. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):063116, 2018.

[200] Nan Chen. Learning nonlinear turbulent dynamics from partial observations via analytically solvable conditional statistics. *Journal of Computational Physics*, 418:109635, 2020.

[201] Pileun Kim, Jonathan Rogers, Jie Sun, and Erik Bollt. Causation entropy identifies sparsity structure for parameter estimation of dynamic systems. *Journal of Computational and Nonlinear Dynamics*, 12(1):011008, 2017.

[202] Abd AlRahman R AlMomani, Jie Sun, and Erik Bollt. How entropic regression beats the outliers problem in nonlinear system identification. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1), 2020.

[203] Jeremie Fish, Alexander DeWitt, Abd AlRahman R AlMomani, Paul J Laurienti, and Erik Bollt. Entropic regression with neurologically motivated applications. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(11), 2021.

[204] Abd AlRahman AlMomani and Erik Bollt. Erfit: Entropic regression fit matlab package, for data-driven system identification of underlying dynamic equations. *arXiv preprint arXiv:2010.02411*, 2020.

[205] Heng Xiao, J-L Wu, J-X Wang, Rui Sun, and CJ Roy. Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach. *Journal of Computational Physics*, 324:115–136, 2016.

[206] Xin-Lei Zhang, Heng Xiao, Xiaodong Luo, and Guowei He. Ensemble Kalman method for learning turbulence models from indirect observation data. *Journal of Fluid Mechanics*, 949:A26, 2022.

[207] Suraj Pawar, Shady E Ahmed, Omer San, and Adil Rasheed. Data-driven recovery of hidden physics in reduced order modeling of fluid flows. *Physics of Fluids*, 32(3):036602, 2020.

[208] Azam Moosavi, Razvan Stefanescu, and Adrian Sandu. Efficient construction of local parametric reduced order models using machine learning techniques. *arXiv preprint arXiv:1511.02909*, 2015.

[209] Omer San and Romit Maulik. Extreme learning machine for reduced order modeling of turbulent geophysical flows. *Physical Review E*, 97(4):042322, 2018.

[210] Andrea Beck and Marius Kurz. A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, 44(1):e202100002, 2021.

[211] Hai-Jun Rong, N Sundararajan, Guang-Bin Huang, and P Saratchandran. Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. *Fuzzy sets and systems*, 157(9):1260–1275, 2006.

[212] TJJ Lombaerts, HO Huisman, QP Chu, Jan A Mulder, and DA Joosten. Nonlinear reconfiguring flight control based on online physical model identification. *Journal of Guidance, Control, and Dynamics*, 32(3):727–748, 2009.

[213] Yannis Kopsinis, Konstantinos Slavakis, and Sergios Theodoridis. Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls. *IEEE Transactions on Signal Processing*, 59(3):936–952, 2010.

[214] Nicholas Kalouptsidis, Gerasimos Mileounis, Behtash Babadi, and Vahid Tarokh. Adaptive algorithms for sparse system identification. *Signal Processing*, 91(8):1910–1919, 2011.

[215] Tianshi Chen, Martin S Andersen, Lennart Ljung, Alessandro Chiuso, and Gianluigi Pillonetto. System identification via sparse multiple kernel-based regularization using sequential convex optimization techniques. *IEEE Transactions on Automatic Control*, 59(11):2933–2945, 2014.

[216] Sriniketh Srinivasan, Julien Billeter, and Dominique Bonvin. Sequential model identification of reaction systems—the missing path between the incremental and simultaneous approaches. *AIChE Journal*, 65(4):1211–1221, 2019.

[217] Georg A Gottwald and Sebastian Reich. Supervised learning from noisy observations: Combining machine-learning techniques with data assimilation. *Physica D: Nonlinear Phenomena*, 423:132911, 2021.

[218] Alexander Wikner, Jaideep Pathak, Brian R Hunt, Istvan Szunyogh, Michelle Girvan, and Edward Ott. Using data assimilation to train a hybrid forecast system that combines machine-learning and knowledge-based components. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(5), 2021.

[219] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[220] Jared Elinger. *Information Theoretic Causality Measures For Parameter Estimation and System Identification*. PhD thesis, Georgia Institute of Technology, 2020.

[221] Jared Elinger and Jonathan Rogers. Causation entropy method for covariate selection in dynamic models. In *2021 American Control Conference (ACC)*, pages 2842–2847. IEEE, 2021.

[222] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[223] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156. Citeseer, 1996.

[224] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[225] Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*, 42(1):164–166, 2013.

[226] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[227] Christopher J Quinn, Negar Kiyavash, and Todd P Coleman. Directed information graphs. *IEEE Transactions on information theory*, 61(12):6887–6909, 2015.

[228] Aaron D Wyner. A definition of conditional mutual information for arbitrary ensembles. *Information and Control*, 38(1):51–59, 1978.

[229] James Massey et al. Causality, feedback and directed information. In *Proc. Int. Symp. Inf. Theory Applic.(ISITA-90)*, pages 303–305, 1990.

[230] Thomas Schreiber. Measuring information transfer. *Physical Review Letters*, 85(2):461, 2000.

[231] Adrián Lozano-Durán, H Jane Bae, and Miguel P Encinar. Causality of energy-containing eddies in wall turbulence. *Journal of Fluid Mechanics*, 882:A2, 2020.

[232] Adrián Lozano-Durán and Gonzalo Arranz. Information-theoretic formulation of dynamical systems: causality, modeling, and control. *Physical Review Research*, 4(2):023195, 2022.

[233] Raul Vicente, Michael Wibral, Michael Lindner, and Gordon Pipa. Transfer entropy—a model-free measure of effective connectivity for the neurosciences. *Journal of computational neuroscience*, 30:45–67, 2011.

[234] Terry Bossomaier, Lionel Barnett, Michael Harré, Joseph T Lizier, Terry Bossomaier, Lionel Barnett, Michael Harré, and Joseph T Lizier. *Transfer entropy*. Springer, 2016.

[235] Jie Sun and Erik M Bollt. Causation entropy identifies indirect influences, dominance of neighbors and anticipatory couplings. *Physica D: Nonlinear Phenomena*, 267:49–57, 2014.

[236] Nicola Branchini, Virginia Aglietti, Neil Dhir, and Theodoros Damoulas. Causal entropy optimization. *arXiv preprint arXiv:2208.10981*, 2022.

[237] Jie Sun, Dane Taylor, and Erik M Bollt. Causal network inference by optimal causation entropy. *SIAM Journal on Applied Dynamical Systems*, 14(1):73–106, 2015.

[238] Richard Ernest Bellman. *Dynamic programming treatment of the traveling salesman problem*. RAND Corporation, 1961.

[239] Michael K Tippett, Richard Kleeman, and Youmin Tang. Measuring the potential utility of seasonal climate predictions. *Geophysical Research Letters*, 31(22), 2004.

[240] Richard Kleeman. Information theory and dynamical system predictability. *Entropy*, 13(3):612–649, 2011.

[241] Michal Branicki and Andrew J Majda. Quantifying uncertainty for predictions with model error in non-Gaussian systems with intermittency. *Nonlinearity*, 25(9):2543, 2012.

[242] Lionel Barnett, Adam B Barrett, and Anil K Seth. Granger causality and transfer entropy are equivalent for Gaussian variables. *Physical Review Letters*, 103(23):238701, 2009.

[243] Ali Shojaie and Emily B Fox. Granger causality: A review and recent advances. *Annual Review of Statistics and Its Application*, 9:289–319, 2022.

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

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

[246] Hermann Haken. Analogy between higher instabilities in fluids and lasers. *Physics Letters A*, 53(1):77–78, 1975.

[247] Edgar Knobloch. Chaos in the segmented disc dynamo. *Physics Letters A*, 82(9):439–440, 1981.

[248] M Gorman, PJ Widmann, and KA Robbins. Nonlinear dynamics of a convection loop: a quantitative comparison of experiment with theory. *Physica D: Nonlinear Phenomena*, 19(2):255–267, 1986.

[249] Neyram Hemati. Strange attractors in brushless DC motors. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 41(1):40–45, 1994.

[250] Kevin M Cuomo and Alan V Oppenheim. Circuit implementation of synchronized chaos with applications to communications. *Physical Review Letters*, 71(1):65, 1993.

[251] Douglas Poland. Cooperative catalysis and chemical chaos: a chemical model for the lorenz equations. *Physica D: Nonlinear Phenomena*, 65(1-2):86–99, 1993.

[252] Stephan I Tzenov. Strange attractors characterizing the osmotic instability. *arXiv preprint arXiv:1406.0979*, 2014.

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

[254] Daniel S Wilks. Effects of stochastic parametrizations in the Lorenz'96 system. *Quarterly Journal of the Royal Meteorological Society*, 131(606):389–407, 2005.

[255] Y Lee and AJ Majda. Multiscale data assimilation and prediction using clustered particle filters. *J Comput Phys*, 2017.

[256] HM Arnold, IM Moroz, and TN Palmer. Stochastic parametrizations and model uncertainty in the Lorenz'96 system. *Phil. Trans. R. Soc. A*, 371(1991):20110479, 2013.

[257] Nan Chen and Andrew J Majda. Beating the curse of dimension with accurate statistics for the fokker–planck equation in complex turbulent systems. *Proceedings of the National Academy of Sciences*, 114(49):12864–12869, 2017.

[258] Wojciech W Grabowski. An improved framework for superparameterization. *Journal of the Atmospheric Sciences*, 61(15):1940–1952, 2004.

[259] David John Gagne, Hannah M Christensen, Aneesh C Subramanian, and Adam H Monahan. Machine learning for stochastic parameterization: Generative adversarial networks in the Lorenz'96 model. *Journal of Advances in Modeling Earth Systems*, 12(3):e2019MS001896, 2020.

[260] Ashesh Chattopadhyay, Pedram Hassanzadeh, and Devika Subramanian. Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, 27(3):373–389, 2020.

[261] Kay Bergemann and Sebastian Reich. A localization technique for ensemble Kalman filters. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 136(648):701–707, 2010.

[262] Jeffrey L Anderson. Exploring the need for localization in ensemble data assimilation using a hierarchical ensemble filter. *Physica D: Nonlinear Phenomena*, 230(1-2):99–111, 2007.

[263] Tijana Janjić, Lars Nerger, Alberta Albertella, Jens Schröter, and Sergey Skachko. On domain localization in ensemble-based Kalman filter algorithms. *Monthly Weather Review*, 139(7):2046–2060, 2011.

[264] Andrew J Majda, Ilya Timofeyev, and Eric Vanden-Eijnden. Systematic strategies for stochastic mode reduction in climate. *Journal of the Atmospheric Sciences*, 60(14):1705–1722, 2003.

[265] Boris Gershgorin, John Harlim, and Andrew J Majda. Improving filtering and prediction of spatially extended turbulent systems with model errors through stochastic parameter estimation. *Journal of Computational Physics*, 229(1):32–57, 2010.

[266] Boris Gershgorin, John Harlim, and Andrew J Majda. Test models for improving filtering with model errors through stochastic parameter estimation. *Journal of Computational Physics*, 229(1):1–31, 2010.

[267] Crispin W Gardiner. Handbook of stochastic methods for physics, chemistry and the natural sciences, vol. 13 of Springer Series in Synergetics, 2004.

[268] Michal Branicki and Andrew J Majda. Dynamic stochastic superresolution of sparsely observed turbulent systems. *Journal of Computational Physics*, 241:333–363, 2013.

[269] Michal Branicki, Andrew J Majda, and Kody J H Law. Accuracy of some approximate Gaussian filters for the Navier-Stokes equation in the presence of model error. *Multiscale Modeling and Simulation*, 2018. Submitted.

[270] Geir Evensen and Peter Jan Van Leeuwen. An ensemble Kalman smoother for nonlinear dynamics. *Monthly Weather Review*, 128(6):1852–1867, 2000.

[271] Uriel Frisch. *Turbulence: the legacy of AN Kolmogorov*. Cambridge university press, 1995.

[272] Ralph H Abraham. Complex dynamical systems. In *Mathematical modelling in science and technology*, pages 82–86. Elsevier, 1984.

[273] Rick Salmon. *Lectures on geophysical fluid dynamics*. Oxford University Press, 1998.

[274] Michael Brin and Garrett Stuck. *Introduction to dynamical systems*. Cambridge University Press, 2002.

[275] TN Palmer. A nonlinear dynamical perspective on climate change. *Weather*, 48(10):314–326, 1993.

[276] Mohammad Farazmand and Themistoklis P Sapsis. Extreme events: Mechanisms and prediction. *Applied Mechanics Reviews*, 71(5), 2019.

[277] William Lahoz, Boris Khattatov, and Richard Ménard. Data assimilation and information. In *Data Assimilation*, pages 3–12. Springer, 2010.

[278] Judith A Curry and Peter J Webster. Climate science and the uncertainty monster. *Bulletin of the American Meteorological Society*, 92(12):1667–1682, 2011.

[279] Timothy DelSole. Predictability and information theory. Part I: Measures of predictability. *Journal of the Atmospheric Sciences*, 61(20):2425–2440, 2004.

[280] Paul N Edwards. Global climate science, uncertainty and politics: Data-laden models, model-filtered data. *Science as Culture*, 8(4):437–472, 1999.

[281] Andrew J Majda and Michal Branicki. Lessons in uncertainty quantification for turbulent dynamical systems. *Discrete & Continuous Dynamical Systems-A*, 32(9):3133–3221, 2012.

[282] Nicolaas G Van Kampen. Stochastic differential equations. *Physics Reports*, 24(3):171–228, 1976.

[283] Ludwig Arnold. Stochastic differential equations. *New York*, 2, 1974.

[284] Peter E Kloeden and Eckhard Platen. Stochastic differential equations. In *Numerical solution of stochastic differential equations*, pages 103–160. Springer, 1992.

[285] Philip E Protter. *Stochastic differential equations*. Springer, 2005.

[286] Kunihiko Taira, Maziar S Hemati, Steven L Brunton, Yiyang Sun, Karthik Duraisamy, Shervin Bagheri, Scott TM Dawson, and Chi-An Yeh. Modal analysis of fluid flows: Applications and outlook. *AIAA Journal*, 58(3):998–1022, 2020.

[287] Xuping Xie, Muhammad Mohebujjaman, Leo G Rebholz, and Traian Iliescu. Data-driven filtered reduced order modeling of fluid flows. *SIAM Journal on Scientific Computing*, 40(3):B834–B857, 2018.

[288] Mickaël D Chekroun and Dmitri Kondrashov. Data-adaptive harmonic spectra and multilayer Stuart-Landau models. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(9):093110, 2017.

[289] Francesco Smarra, Achin Jain, Tullio De Rubeis, Dario Ambrosini, Alessandro D'Innocenzo, and Rahul Mangharam. Data-driven model predictive control using random forests for building energy optimization and climate control. *Applied Energy*, 226:1252–1272, 2018.

[290] Yang Tang, Jürgen Kurths, Wei Lin, Edward Ott, and Ljupco Kocarev. Introduction to focus issue: When machine learning meets complex systems: Networks, chaos, and nonlinear dynamics. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(6), 2020.

[291] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[292] Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, and Karen Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.

[293] Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model. *Journal of Computational Science*, 44:101171, 2020.

[294] Rossella Arcucci, Jiangcheng Zhu, Shuang Hu, and Yi-Ke Guo. Deep data assimilation: integrating deep learning with data assimilation. *Applied Sciences*, 11(3):1114, 2021.

[295] Caterina Buizza, César Quilodrán Casas, Philip Nadler, Julian Mack, Stefano Marrone, Zainab Titus, Clémence Le Cornec, Evelyn Heylen, Tolga Dur, Luis Baca Ruiz, et al. Data learning: Integrating data assimilation and machine learning. *Journal of Computational Science*, 58:101525, 2022.

[296] Georg A Gottwald and Sebastian Reich. Combining machine learning and data assimilation to forecast dynamical systems from noisy partial observations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(10), 2021.

[297] Robin C Gilbert, Michael B Richman, Theodore B Trafalis, and Lance M Leslie. Machine learning methods for data assimilation. *Computational Intelligence in Architecturing Complex Engineering Systems*, pages 105–112, 2010.

[298] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[299] Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

[300] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. *Advances in Neural Information Processing Systems*, 30, 2017.

[301] Alban Farchi, Marc Bocquet, Patrick Laloyaux, Massimo Bonavita, and Quentin Malartic. A comparison of combined data assimilation and machine learning methods for offline and online model error correction. *Journal of Computational Science*, 55:101468, 2021.

[302] Alban Farchi, Marcin Chrust, Marc Bocquet, Patrick Laloyaux, and Massimo Bonavita. Online model error correction with neural networks in the incremental 4D-Var framework. *Journal of Advances in Modeling Earth Systems*, 15(9):e2022MS003474, 2023.

[303] Marc Bocquet, Alban Farchi, and Quentin Malartic. Online learning of both state and dynamics using ensemble Kalman filters. *arXiv preprint arXiv:2006.03859*, 2020.

[304] Sibo Cheng and Mingming Qiu. Observation error covariance specification in dynamical systems for data assimilation using recurrent neural networks. *Neural Computing and Applications*, 34(16):13149–13167, 2022.

[305] William Vega-Brown, Abraham Bachrach, Adam Bry, Jonathan Kelly, and Nicholas Roy. Cello: A fast algorithm for covariance estimation. In *2013 IEEE International Conference on Robotics and Automation*, pages 3160–3167. IEEE, 2013.

[306] Katherine Liu, Kyel Ok, William Vega-Brown, and Nicholas Roy. Deep inference for covariance estimation: Learning Gaussian noise models for state estimation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1436–1443. IEEE, 2018.

[307] Pierre Boudier, Anthony Fillion, Serge Gratton, and Selime Gürol. DAN–An optimal data assimilation framework based on machine learning recurrent networks. *arXiv preprint arXiv:2010.09694*, 2020.

[308] Said Ouala, Ronan Fablet, Cédric Herzet, Bertrand Chapron, Ananda Pascual, Fabrice Collard, and Lucile Gaultier. Neural network based Kalman filters for the spatio-temporal interpolation of satellite-derived sea surface temperature. *Remote Sensing*, 10(12):1864, 2018.

[309] Georgy E Manucharyan, Lia Siegelman, and Patrice Klein. A deep learning approach to spatiotemporal sea surface height interpolation and estimation of deep currents in geostrophic ocean turbulence. *Journal of Advances in Modeling Earth Systems*, 13(1):e2019MS001965, 2021.

[310] Yuming Chen, Daniel Sanz-Alonso, and Rebecca Willett. Autodifferentiable ensemble kalman filters. *SIAM Journal on Mathematics of Data Science*, 4(2):801–833, 2022.

[311] Nan Chen and Andrew J Majda. Filtering nonlinear turbulent dynamical systems through conditional Gaussian statistics. *Monthly Weather Review*, 144(12):4885–4917, 2016.

[312] Robert S Liptser and Albert N Shiryaev. *Statistics of random processes II: Applications*, volume 6. Springer Science & Business Media, 2013.

[313] Nan Chen, Andrew J Majda, and Dimitrios Giannakis. Predicting the cloud patterns of the madden-julian oscillation through a low-order nonlinear stochastic model. *Geophysical Research Letters*, 41(15):5612–5619, 2014.

[314] Nan Chen, Shubin Fu, and Georgy E Manucharyan. An efficient and statistically accurate Lagrangian data assimilation algorithm with applications to discrete element sea ice models. *Journal of Computational Physics*, 455:111000, 2022.

[315] Nan Chen, Andrew J Majda, and Xin T Tong. Information barriers for noisy lagrangian tracers in filtering random incompressible flows. *Nonlinearity*, 27(9):2133, 2014.

[316] Nan Chen and Andrew J Majda. Model error in filtering random compressible flows utilizing noisy lagrangian tracers. *Monthly Weather Review*, 144(11):4037–4061, 2016.

[317] Nan Chen, Evelyn Lunasin, and Stephen Wiggins. Lagrangian descriptors with uncertainty. *arXiv preprint arXiv:2307.04006*, 2023.

[318] Nan Chen and Andrew J Majda. Filtering the stochastic skeleton model for the Madden–Julian oscillation. *Monthly Weather Review*, 144(2):501–527, 2016.

[319] Nan Chen, Andrew J Majda, CT Sabeerali, and RS Ajayamohan. Predicting monsoon intraseasonal precipitation using a low-order nonlinear stochastic model. *Journal of Climate*, 2018.

[320] Shane R Keating, Andrew J Majda, and K Shafer Smith. New methods for estimating ocean eddy heat transport using satellite altimetry. *Monthly Weather Review*, 140(5):1703–1722, 2012.

[321] Andrew J Majda and Marcus J Grote. Mathematical test models for superparametrization in anisotropic turbulence. *Proceedings of the National Academy of Sciences*, 106(14):5470–5474, 2009.

[322] Ian Grooms and Andrew J Majda. Stochastic superparameterization in quasi-geostrophic turbulence. *Journal of Computational Physics*, 271:78–98, 2014.

[323] Andrew J Majda and Ian Grooms. New perspectives on superparameterization for geophysical turbulence. *Journal of Computational Physics*, 271:60–77, 2014.

[324] Andrew J Majda, Di Qi, and Themistoklis P Sapsis. Blended particle filters for large-dimensional chaotic dynamical systems. *Proceedings of the National Academy of Sciences*, 111(21):7511–7516, 2014.

[325] Quanling Deng, Nan Chen, Samuel N Stechmann, and Jiuhua Hu. LEMDA: A Lagrangian-Eulerian multiscale data assimilation framework. *arXiv preprint arXiv:2401.18048*, 2024.

[326] Andrew J Majda, Ilya Timofeyev, and Eric Vanden Eijnden. Models for stochastic climate prediction. *Proceedings of the National Academy of Sciences*, 96(26):14687–14691, 1999.

[327] Andrew J Majda, Ilya Timofeyev, and Eric Vanden Eijnden. A mathematical framework for stochastic climate models. *Communications on Pure and Applied Mathematics*, 54(8):891–974, 2001.

[328] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[329] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[330] Xue Ying. An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, volume 1168, page 022022. IOP Publishing, 2019.

[331] Edward N Lorenz. Formulation of a low-order model of a moist general circulation. *Journal of the atmospheric sciences*, 41(12):1933–1945, 1984.

[332] Edward N Lorenz. Irregularity: a fundamental property of the atmosphere. *Tellus A: Dynamic Meteorology and Oceanography*, 36(2):98–110, 1984.

[333] Mickaël D Chekroun, Honghu Liu, and Shouhong Wang. *Stochastic parameterizing manifolds and non-Markovian reduced equations: stochastic manifolds for nonlinear SPDEs II*. Springer, 2014.

[334] Mickaël D Chekroun and Honghu Liu. Post-processing finite-horizon parameterizing manifolds for optimal control of nonlinear parabolic PDEs. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1411–1416. IEEE, 2016.

[335] David Kelly, Andrew J Majda, and Xin T Tong. Concrete ensemble Kalman filters with rigorous catastrophic filter divergence. *Proceedings of the National Academy of Sciences*, 112(34):10589–10594, 2015.

[336] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation: methods, algorithms, and applications*. SIAM, 2016.

[337] Edward Ott, Brian R Hunt, Istvan Szunyogh, Matteo Corazza, Eugenia Kalnay, DJ Patil, James A Yorke, Aleksey V Zimin, and Eric J Kostelich. Exploiting local low dimensionality of the atmospheric dynamics for efficient ensemble Kalman filtering. *arXiv preprint physics/0203058*, 3, 2002.

[338] Larry R Medsker, Lakhmi Jain, et al. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[339] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[340] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[341] Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3), 2021.

[342] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10):2318–2331, 2017.

[343] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

[344] Karthik Kashinath, M Mustafa, Adrian Albert, JL Wu, C Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, R Wang, Ashesh Chattopadhyay, A Singh, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

[345] Xu-Hui Zhou, Jiequn Han, and Heng Xiao. Learning nonlocal constitutive models with neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113927, 2021.

[346] Jiequn Han, Xu-Hui Zhou, and Heng Xiao. An equivariant neural operator for developing nonlocal tensorial constitutive models. *Journal of Computational Physics*, 488:112243, 2023.

[347] Xu-Hui Zhou, Zhuo-Ran Liu, and Heng Xiao. Bi-eqno: Generalized approximate bayesian inference with an equivariant neural operator framework. *arXiv preprint arXiv:2410.16420*, 2024.

[348] George Neofotistos, Marios Mattheakis, Georgios D Barmparis, Johanne Hizanidis, Giorgos P Tsironis, and Efthimios Kaxiras. Machine learning with observers predicts complex spatiotemporal behavior. *Frontiers in Physics*, 7:24, 2019.

[349] Hector Vargas Alvarez, Gianluca Fabiani, Nikolaos Kazantzis, Ioannis G Kevrekidis, and Constantinos Siettos. Nonlinear discrete-time observers with physics-informed neural networks. *Chaos, Solitons & Fractals*, 186(C), 2024.

[350] Bernard O Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[351] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied Koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4), 2012.

[352] Igor Mezić. Analysis of fluid flows via spectral properties of the Koopman operator. *Annual review of fluid mechanics*, 45(1):357–378, 2013.

[353] Shaowu Pan and Karthik Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19(1):480–509, 2020.

[354] Samuel E Otto and Clarence W Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):59–87, 2021.

[355] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern Koopman theory for dynamical systems. *SIAM Review*, 64(2):229–340, 2022.

[356] Shaowu Pan, Eurika Kaiser, Brian M de Silva, J Nathan Kutz, and Steven L Brunton. Pykoopman: a python package for data-driven approximation of the Koopman operator. *arXiv preprint arXiv:2306.12962*, 2023.

[357] Matthew J Colbrook, Igor Mezić, and Alexei Stepanenko. Limits and powers of Koopman learning. *arXiv preprint arXiv:2407.06312*, 2024.

[358] Nan Chen, Andrew J Majda, and Xin T Tong. Noisy lagrangian tracers for filtering random rotating compressible flows. *Journal of Nonlinear Science*, 25(3):451–488, 2015.

[359] Francis J Anscombe. Graphs in statistical analysis. *The american statistician*, 27(1):17–21, 1973.

[360] Robert F Ling. Residuals and influence in regression, 1984.

[361] Nan Chen and Yingda Li. BAMCAFE: A Bayesian machine learning advanced forecast ensemble method for complex turbulent systems with partial observations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(11), 2021.

[362] Jared Elinger. *Information Theoretic Causality Measures For Parameter Estimation and System Identification.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2021.

[363] Alireza Karimi and Mark R Paul. Extensive chaos in the Lorenz-96 model. *Chaos: An interdisciplinary journal of nonlinear science*, 20(4), 2010.

[364] Chester F Ropelewski and Michael S Halpert. Global and regional scale precipitation patterns associated with the El Niño/Southern Oscillation. *Monthly Weather Review*, 115(8):1606–1626, 1987.

[365] Michael J McPhaden, Stephen E Zebiak, and Michael H Glantz. ENSO as an integrating concept in earth science. *Science*, 314(5806):1740–1745, 2006.

[366] Mojib Latif, D Anderson, T Barnett, M Cane, R Kleeman, A Leetmaa, J O'Brien, A Rosati, and E Schneider. A review of the predictability and prediction of enso. *Journal of Geophysical Research: Oceans*, 103(C7):14375–14393, 1998.

[367] J David Neelin, David S Battisti, Anthony C Hirst, Fei-Fei Jin, Yoshinobu Wakata, Toshio Yamagata, and Stephen E Zebiak. ENSO theory. *Journal of Geophysical Research: Oceans*, 103(C7):14261–14290, 1998.

[368] Nan Chen and Xianghui Fang. A simple multiscale intermediate coupled stochastic model for El Niño diversity and complexity. *Journal of Advances in Modeling Earth Systems*, 15(4):e2022MS003469, 2023.

[369] Nan Chen, Xianghui Fang, and Jin-Yi Yu. A multiscale model for El Niño complexity. *npj Climate and Atmospheric Science*, 5(1):1–13, 2022.

[370] Tao Geng, Wenju Cai, and Lixin Wu. Two types of ENSO varying in tandem facilitated by nonlinear atmospheric convection. *Geophysical Research Letters*, 47(15):e2020GL088784, 2020.

[371] Benjamin Sanderse, Panos Stinis, Romit Maulik, and Shady E Ahmed. Scientific machine learning for closure models in multiscale problems: A review. *arXiv preprint arXiv:2403.02913*, 2024.

[372] Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to infer unresolved scale parametrization. *Philosophical Transactions of the Royal Society A*, 379(2194):20200086, 2021.

[373] Yuming Chen, Daniel Sanz-Alonso, and Rebecca Willett. Reduced-order autodifferentiable ensemble Kalman filters. *Inverse Problems*, 39(12):124001, 2023.

[374] Chuanqi Chen, Nan Chen, Yinling Zhang, and Jin-Long Wu. CGKN: A deep learning framework for modeling complex dynamical systems and efficient data assimilation. *Journal of Computational Physics*, 532:113950, 2025.

[375] Matthew O. Williams, Clarence W. Rowley, and Ioannis G. Kevrekidis. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015.

[376] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

[377] Felix Dietrich, Thomas N Thiem, and Ioannis G Kevrekidis. On the Koopman operator of algorithms. *SIAM Journal on Applied Dynamical Systems*, 19(2):860–885, 2020.

[378] Hannah Lu and Daniel M Tartakovsky. Prediction accuracy of dynamic mode decomposition. *SIAM Journal on Scientific Computing*, 42(3):A1639–A1662, 2020.

[379] Petar Bevanda, Stefan Sosnowski, and Sandra Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021.

[380] Hannah Lu and Daniel M Tartakovsky. Extended dynamic mode decomposition for inhomogeneous problems. *Journal of Computational Physics*, 444:110550, 2021.

[381] Zhongrui Wang, Nan Chen, and Di Qi. A closed-form nonlinear data assimilation algorithm for multi-layer flow fields. *arXiv preprint arXiv:2412.11042*, 2024.

[382] Ming Cheng, Peng Wang, and Daniel M Tartakovsky. Efficient quadratures for high-dimensional Bayesian data assimilation. *Journal of Computational Physics*, 506:112945, 2024.

[383] Xin T Tong and Matthias Morzfeld. Localized ensemble Kalman inversion. *Inverse Problems*, 39(6):064002, 2023.

[384] Shuigen Liu, Sebastian Reich, and Xin T Tong. Dropout ensemble Kalman inversion for high dimensional inverse problems. *arXiv preprint arXiv:2308.16784*, 2023.

[385] David Vishny, Matthias Morzfeld, Kyle Gwirtz, Eviatar Bach, Oliver RA Dunbar, and Daniel Hodyss. High-dimensional covariance estimation from a small number of samples. *Journal of Advances in Modeling Earth Systems*, 16(9):e2024MS004417, 2024.

[386] Jeffrey L Anderson and Stephen L Anderson. A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly weather review*, 127:18, 1999.

[387] Gregory Gaspari and Stephen E Cohn. Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125(554):723–757, 1999.

[388] Adrian E Gill. Some simple solutions for heat-induced tropical circulation. *Quarterly Journal of the Royal Meteorological Society*, 106(449):447–462, 1980.

[389] Taroh Matsuno. Quasi-geostrophic motions in the equatorial area. *Journal of the Meteorological Society of Japan. Ser. II*, 44(1):25–43, 1966.

[390] Geoffrey K Vallis. Geophysical fluid dynamics: whence, whither and why? *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2192):20160140, 2016.

[391] Fei-Fei Jin. An equatorial ocean recharge paradigm for enso. part ii: A stripped-down coupled model. *Journal of the Atmospheric Sciences*, 54(7):830–847, 1997.