AdaKoop: Efficient Modeling of Nonlinear Dynamics from Nonstationary Data Streams with Koopman Operator Regression

Naoki Chihara
SANKEN, The University of Osaka
Osaka, Japan
naoki88@sanken.osaka-u.ac.jp

Ren Fujiwara
SANKEN, The University of Osaka
Osaka, Japan
r-fujiwr@sanken.osaka-u.ac.jp

Yasuko Matsubara
SANKEN, The University of Osaka
Osaka, Japan
yasuko@sanken.osaka-u.ac.jp

Yasushi Sakurai
SANKEN, The University of Osaka
Osaka, Japan
yasushi@sanken.osaka-u.ac.jp

Abstract

Real-time data analysis requires the ability to accurately and adaptively address nonlinear dynamics in a nonstationary data stream while preserving computational efficiency. However, nonlinear dynamics are so complex that capturing dynamically changing nonlinear patterns and utilizing them for downstream tasks under strict time constraints is nontrivial. To bridge the gap between nonlinear complexity and computational tractability, this study applies Koopman operator theory, which states that nonlinear dynamics can be represented as linear transitions in an infinite-dimensional space. Building upon finite-dimensional approximations of this operator, we present AdaKoop, an efficient streaming algorithm for modeling nonlinear dynamics over nonstationary data streams. Our approach utilizes a probabilistic framework grounded in Koopman operator theory, treating both raw observations and reproducing kernel Hilbert space (RKHS) features as emissions from latent vectors. This dual-view formulation allows nonlinear dynamics to be expressed as a tractable linear system. Therefore, AdaKoop enables the efficient and stable modeling of nonlinear dynamics in a streaming fashion, avoiding the prohibitive computational costs of iterative nonlinear optimization. Furthermore, to address nonstationarity in data streams, AdaKoop adaptively detects the switching of patterns via statistical hypothesis testing for abrupt pattern shifts and incrementally updates model parameters to handle continuous changes. Extensive experiments on a total of 71 practical benchmark datasets across various domains demonstrate that AdaKoop outperforms state-of-the-art methods in terms of real-time forecasting accuracy and computational efficiency.

CCS Concepts

• **Mathematics of computing** → **Time series analysis**; **Probabilistic inference problems**; • **Information systems** → **Data stream mining**; • **Computing methodologies** → **Kernel methods**.

KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817851

Keywords

Time series, Stream processing, Dynamical system, Koopman operator theory, Reproducing kernel Hilbert space

ACM Reference Format:
Naoki Chihara, Ren Fujiwara, Yasuko Matsubara, and Yasushi Sakurai. 2026. AdaKoop: Efficient Modeling of Nonlinear Dynamics from Nonstationary Data Streams with Koopman Operator Regression. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/3770855.3817851

1 Introduction

With the rapid worldwide digitalization, massive volumes of time series data are collected and processed sequentially, calling for efficient and precise stream processing capabilities across various applications, such as precipitation nowcasting [17] and real-time wearable health monitoring [28]. Unlike batch learning settings, where all the observed data is available in advance, these environments involve potentially semi-infinite data streams. The unbounded nature of these data streams makes it impossible to store the entire history, particularly in resource-constrained edge computing environments [41]. This requires performing downstream tasks (e.g., real-time forecasting) in a single pass over incoming data while adapting rapidly to current dynamic patterns.

However, achieving such adaptive processing remains a challenge because real-world data streams often exhibit intrinsic complexities that hinder efficient real-time data analysis under strict latency constraints. Specifically, we mainly encounter two challenges:
(a) Nonlinear dynamics. Real-world data streams inherently exhibit complex nonlinear dynamics [23]. For example, atmospheric weather patterns are governed by the Lorenz system [40], which is a typical nonlinear dynamical system. In healthcare, variability patterns observed in physiological signals, such as the electrocardiogram (ECG), exhibit deterministic chaotic behavior [24]. Nonlinear dynamics are too complex for simple linear models, so a more expressive model is required. Recently, the data-driven discovery of governing nonlinear dynamics from data has attracted significant interest, and various approaches have been proposed, including polynomial regressions [4, 7, 18] and recurrent neural networks (RNNs) [9, 10]. However, most previous methods involve computationally intensive optimization or exhibit high sensitivity to noise, making them suboptimal for data stream processing.

(b) Nonstationarity. The statistical properties of a data stream can change continuously and abruptly over time due to external influences or shifts in underlying mechanisms. For example, in wearable health monitoring, physiological signal patterns may change gradually with circadian rhythms but can shift abruptly when the wearer changes activities (e.g., from resting to exercising) or when the sensor is re-positioned. This variability further exacerbates data stream processing because future values may exhibit patterns different from those observed in the past. Specifically, even if a model can learn patterns accurately in the early stages, it will inevitably degrade as time-changing patterns emerge. Based on the above discussion, our problem is summarized as follows:

> *How can we estimate complex nonlinear dynamics while remaining computationally efficient enough to quickly adapt to both continuous and abrupt changes in a nonstationary data stream?*

In this paper, we propose an efficient streaming algorithm called AdaKoop, designed for modeling nonlinear dynamics in nonstationary data streams. To capture nonlinear dynamics efficiently, we first introduce a probabilistic augmented kernel state space model constructed based on Koopman operator theory, defined on a reproducing kernel Hilbert space (RKHS). The Koopman operator provides a linear representation of nonlinear dynamical systems by lifting the state space to an infinite-dimensional space. Our model treats both raw observations and RKHS features as emissions from a shared latent linear dynamical system; therefore, it enables complex nonlinear dynamics to be expressed as a tractable linear system. Thus, AdaKoop enables efficient and stable modeling of nonlinear dynamics via closed-form updates. To further improve the efficiency of the estimation, we initialize the model parameters based on Koopman operator regression and subsequently refine them using the expectation-maximization (EM) algorithm. This initialization serves as a warm start, helping to avoid undesired local optima and accelerate convergence. In addition, our streaming algorithm performs both abrupt change detection based on statistical hypothesis testing and continuous model parameter updates to adaptively handle nonstationary data streams, making it highly scalable for semi-infinite data streams.

Contributions. Our main contributions are summarized as follows:

- We introduce a probabilistic dual-view state space model that treats both raw observations and RKHS features as emissions from a shared latent linear system, thereby achieving stable and efficient modeling of nonlinear dynamics.
- We present AdaKoop, an efficient streaming algorithm with constant time complexity per process, independent of the length of a data stream (see Lemma 2), that handles time-varying nonlinear dynamics in a data stream. It employs statistical hypothesis testing to adaptively switch models and utilizes closed-form updates for stream processing.
- Extensive experiments on 71 practical benchmark datasets show that AdaKoop outperforms state-of-the-art methods in both forecasting accuracy and computational efficiency.

Reproducibility. Our source code is available at this repository: https://github.com/C-Naoki/AdaKoop.

Table 1: Capabilities of approaches.

	KOT				TSF			DSP		
	FSDMD [54]	EDMD [58]	WDMD [59]	Kostic et al. [37]	PAttn [55]	Koopa [38]	OneNet [56]	sKAF [22]	ModePlait [11]	**AdaKoop**
Online algorithm			✓				✓	✓	✓	✓
Forecasting					✓	✓	✓	✓	✓	✓
Nonlinearity		✓		✓		✓	✓	✓	✓	✓
Nonstationarity	✓		✓			✓	✓		✓	✓
Infinite feature space		✓		✓				✓		✓
Linear representation	✓	✓	✓	✓		✓				✓

2 Related Work

Here, we provide an overview of investigations related to our work. Table 1 summarizes six relative advantages of AdaKoop.

Koopman operator theory. Koopman operator theory (KOT) [6, 34] has been developed over decades to analyze and understand nonlinear dynamical systems without any explicit prior knowledge of their governing temporal dynamics [3, 30, 37]. According to KOT, every deterministic nonlinear dynamical system is represented by a linear yet infinite-dimensional operator without losing information. One advantage of using linear operators is that they enable spectral decomposition, which provides insights into the behavior of nonlinear dynamical systems. Dynamic mode decomposition (DMD) is a representative method for approximating the Koopman operator. DMD was originally proposed as a tool for diagnosing fluid flows [52], but its reliance on linear monomials as basis functions in the feature space restricts its ability to model nonlinear dynamics. Subsequent research has explored various extensions to broaden its range of applications [50, 53]. For example, FSDMD [54] and Windowed DMD (WDMD) [59] address time-varying linear systems through distinct methodologies. EDMD [58] addresses the limitations of linear monomials by mapping data into a high-dimensional space defined by user-specified nonlinear basis functions. Recently, several methods aim to learn the Koopman operators on a reproducing kernel Hilbert space (RKHS) [33, 36, 43]. A framework to learn the Koopman operator in an RKHS using finite temporally sampled data was presented in [37]. However, they assume that the input data is generated by an autonomous dynamical system, meaning that they cannot handle ever-changing nonlinear dynamics in a nonstationary data stream.

Data stream processing. Considering the real-world scenario where a large number of time series data is generated sequentially, data stream processing (DSP) has been increasingly recognized for its practicality and efficiency [20]. In particular, as underlying patterns are likely to shift over time, methods incapable of adapting to these changes often suffer from degraded accuracy. Thus, DSP has proved greatly significant to the machine learning and database communities [1, 13, 15, 19, 25, 27, 46]. ModePlait [11] is the latest streaming algorithm for discovering time-changing causal relationships and forecasting future values while monitoring transitions of dynamic patterns. However, it remains challenging to estimate highly nonlinear dynamics. Although Streaming KAF (sKAF) [22]

utilizes a streaming algorithm to reduce computational complexity, its update rule implicitly assumes that each data point is obtained from a static environment. Moreover, it directly generates future values from current data without focusing on the sequential nature of time series, and its forecasting performance can be unstable.

Time series forecasting. Time series forecasting (TSF) is one of the most important tasks in time series analysis. Autoregressive integrated moving average (ARIMA) [5] and linear dynamical system (LDS) [29] are classical statistical methods for TSF. In recent years, numerous TSF methods have benefited from deep learning approaches [45, 49, 60]. For example, PAttn [55] is a state-of-the-art, simple linear-based method. Koopa [38] utilizes Koopman operator theory to learn nonstationary time series. However, most of these approaches focus only on offline optimization and cannot adaptively handle time-changing patterns in a nonstationary data stream. In contrast, OneNet [56] is an online learning approach that addresses transient environmental changes by dynamically updating two complementary models. While the online ensembling approach decreases the forecasting error, it also increases the number of parameters and computational time, limiting its practical applicability in a streaming setting.

3 Model Formulation

Here, we formulate the problem addressed in this work and then present the AdaKoop model designed to tackle this challenge.

3.1 Problem Definition

Notation. The main symbols utilized in this paper are in Table 4. We define $[m:n] = \{m, \dots, n\}$ and $[n] = \{1, \dots, n\}$ for $n, m \in \mathbb{N}$. Let $\mathcal{M}$ be an observation space, and $\mathcal{H}$ be a reproducing kernel Hilbert space (RKHS) with kernel function $k : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ and feature map $\phi : \mathcal{M} \to \mathcal{H}$ [2], such that $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}}$ for $\mathbf{x}, \mathbf{y} \in \mathcal{M}$. The Moore-Aronszajn theorem guarantees that a symmetric, positive kernel function k provides a unique RKHS.

In this paper, we focus on efficiently modeling nonlinear dynamics from nonstationary data streams and using the learned models for real-time forecasting. Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_{t_c}, ...\}$ be a semi-infinite multivariate data stream where t_c is the current time and $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional observation at time t. In our model, we assume that observation data $\mathbf{x}_t$ behaves as the following time-varying discrete dynamical systems,

$$\mathbf{x}_{t+1} = \boldsymbol{f}_t(\mathbf{x}_t) + \boldsymbol{\omega}_t, \quad t \in \mathbb{N}, \tag{1}$$

where $\boldsymbol{f}_t$ is a nonlinear state-transition function describing the dynamics at time point t, and $\boldsymbol{\omega}_t$ denotes an i.i.d. zero-mean noise term. Nonstationarity mainly arises from the time-dependency of $\boldsymbol{f}_t$. Thus, we tackle adaptively estimating nonlinear time-varying dynamical systems $\boldsymbol{f}_t$ from a data stream and forecast an ℓ_s-steps-ahead future value, i.e., $\mathbf{x}_{t_c+\ell_s}$ sequentially. Consequently, the problem addressed in our work is summarized as follows:

Problem 1. ***Given*** *a semi-infinite multivariate data stream* $\mathbf{X}$*, which consists of* d*-dimensional vector* $\mathbf{x}_t$ *generated sequentially, i.e.,* $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_{t_c}, ...\}$*, where* t_c *is the current time,*

- ***Capture*** *complex nonlinear dynamic patterns over time,*
- ***Forecast*** ℓ_s*-steps-ahead future values, i.e.,* $\mathbf{x}_{t_c+\ell_s}$*,*

continuously and quickly, in a streaming fashion.

3.2 Proposed Model – AdaKoop

We now explain our proposed model in detail. To achieve our goal, we require AdaKoop model to be capable of the following.

- *Dual-view kernelized system*: We formulate a probabilistic augmented state space model that treats both raw observations and RKHS features as emissions from a shared latent linear dynamical system.
- *Switching structures*: Our model represents the underlying time-varying system $\boldsymbol{f}_t$ as a switching dynamical system composed of multiple distinct local patterns.

3.2.1 Dual-view kernelized system. We begin with a simple case in which we have only a single dynamic pattern in a data stream. To capture nonlinear dynamics within a linear state-space framework, we leverage the concept of the Koopman operator, which lifts the state into a higher-dimensional feature space where the dynamics become linear. Let $\psi : \mathbb{R}^d \to \mathbb{R}^m$ be a feature map associated with an RKHS, approximating the infinite-dimensional lifting. We define an augmented observation vector $\mathbf{y}_t \in \mathbb{R}^{d+m}$ combining the dual views of the raw data and its feature representation

$$\mathbf{y}_t = \begin{bmatrix} \mathbf{x}_t \\ \psi(\mathbf{x}_t) \end{bmatrix}.$$

Ideally, $\psi(\mathbf{x}_t)$ is functionally dependent on $\mathbf{x}_t$, but modeling this deterministic constraint strictly within a generative probabilistic framework leads to a degenerate likelihood, making inference numerically unstable. To address this, we adopt a relaxation strategy: we treat the augmented vector $\mathbf{y}_t$ as a noisy observation emitted from a low-dimensional latent vector $\mathbf{z}_t \in \mathbb{R}^r$. This formulation posits that the dominant dynamics evolve on a low-dimensional manifold [42]. Furthermore, this relaxation allows the probabilistic model to absorb approximation errors arising from the finite-dimensional truncation of the Koopman operator into the observation noise. Consequently, we describe the single dynamic pattern using the following equation:

Model 1. *Let* $\mathbf{z}_t \in \mathbb{R}^r$ *be the latent vector at time* t*, and* $\mathbf{y}_t \in \mathbb{R}^{d+m}$ *the augmented observation vectors at time* t*. The following equation governs the single nonlinear dynamic pattern.*

$$\begin{aligned} \mathbf{z}_{t+1} &= \mathbf{A}\mathbf{z}_t + \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \\ \mathbf{y}_t &= \mathbf{H}\mathbf{z}_t + \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \end{aligned} \tag{2}$$

with initial conditions $\mathbf{z}_1 \sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{P}_0)$ *In this equation,* $\mathbf{A} \in \mathbb{R}^{r \times r}$ *is a transition matrix and* $\mathbf{H} = [\mathbf{C}^\top \ \mathbf{W}^\top]^\top \in \mathbb{R}^{(d+m) \times r}$ *is an augmented observation matrix, where* $\mathbf{C} \in \mathbb{R}^{d \times r}$ *maps the latent state to the raw observation and* $\mathbf{W} \in \mathbb{R}^{m \times r}$ *maps it to the feature space. The process noise covariance is* $\mathbf{Q} \in \mathbb{R}^{r \times r}$ *and the observation noise covariance is given by* $\mathbf{R} = \mathrm{BlockDiag}(\mathbf{R}_\mathbf{x}, \mathbf{R}_\psi) \in \mathbb{R}^{(d+m) \times (d+m)}$*, where* $\mathbf{R}_\mathbf{x} \in \mathbb{R}^{d \times d}$ *represents measurement noise and* $\mathbf{R}_\psi \in \mathbb{R}^{m \times m}$ *accommodates the kernel approximation residuals.*

Here, we enforce this block-diagonal structure by assuming statistical independence between the measurement noise in $\mathbf{x}_t$ and the approximation residuals in $\psi(\mathbf{x}_t)$. This structural constraint decouples the two error sources, thereby stabilizing the inference and preventing the model from overfitting to the deterministic dependency between the views. In addition, while Model 1 is formulated as a linear system, it can accurately capture a wide range

of complex nonlinear dynamics, which is supported by Koopman operator theory. Consequently, we have the following:

Definition 1 (Dual-view kernelized system: DKS). *Let $\boldsymbol{\theta}$ be parameters of a dual-view kernelized system for a single nonlinear dynamic pattern, i.e., $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{Q}, \mathbf{C}, \mathbf{W}, \mathbf{R}_x, \mathbf{R}_\psi, \boldsymbol{\mu}_0, \mathbf{P}_0\}$.*

3.2.2 Switching structures. Model 1 is suitable for summarizing time series data generated by a time-invariant system. However, considering the real-world scenario where dynamic patterns are ever-changing based on $\boldsymbol{f}_t$, its ability is still insufficient for data stream processing. To overcome this limitation, we introduce a more comprehensive model. We assume that time-varying systems are approximated by discrete switching dynamical systems. This assumption is generally valid because the dynamic patterns do not frequently change over time. Then, the data stream $\mathbf{X}$ is summarized by the model set $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R\}$, where R is the number of models. Consequently, we have the following:

Definition 2 (Dual-view kernelized system set). *Let $\boldsymbol{\Theta}$ be a parameter set of multiple dual-view kernelized systems, i.e., $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_R\}$, which describes distinct nonlinear dynamic patterns in a data stream.*

4 Optimization Algorithm

In this section, we present an efficient algorithm with which we estimate the model set $\boldsymbol{\Theta}$ describing a semi-infinite data stream $\mathbf{X}$ continuously. We first introduce a static optimization to estimate model parameters from time series data, and then present a streaming algorithm for monitoring current patterns and forecasting in a streaming fashion based on the static optimization.

4.1 Static Optimization

Here, we describe the static optimization to estimate the best model parameter $\boldsymbol{\theta}$ for a time series $\mathbf{X} \in \mathbb{R}^{d \times T}$. Figure 1 illustrates the overall framework and Algorithm 2 (see Appendix C) shows its estimation procedure. The static optimization consists of the following three components:

- *Feature space basis orthogonalization*: We construct a sparse finite-dimensional feature space from the RKHS to ensure computational stability and efficiency.
- *Regression-based initialization*: Spectral initialization of the model parameters via reduced rank regression (RRR), which interprets the system dynamics as a finite-rank approximation of the Koopman operator.
- *Probabilistic refinement*: We refine the initialized parameters by maximizing the marginal likelihood using the EM algorithm, treating the latent states as hidden variables.

4.1.1 Feature space basis orthogonalization. In this section, we describe how to efficiently optimize our proposed model. Generally, kernel-based approaches tend to be computationally expensive as the dataset size increases. This property is the major drawback for data stream processing [15]. Additionally, linearly dependent feature vectors in the RKHS may lead to redundant representations and numerical instability. Hence, we use the criterion of whether the model should incorporate an observed data point. According to the representer theorem [31], the optimal solution is given by a linear combination of $\{k(\cdot, \mathbf{x}_t)\}_{t=1}^T$, i.e., $\boldsymbol{f}(\mathbf{x}) = \sum_{t=1}^T \beta_t k(\mathbf{x}, \mathbf{x}_t)$. This implies that if $\phi(\mathbf{x}_t)$ lies approximately in the span of $\{\phi(\mathbf{x}_\tau)\}_{\tau<t}$, then adding $\mathbf{x}_t$ does not enlarge the hypothesis subspace and the corresponding coefficient β_t can be set to zero. Motivated by this observation, we maintain a dictionary of selected observations $\mathcal{D}_t = \{\mathbf{x}_\tau \mid \tau \in \mathcal{I}_t\}$, where $\mathcal{I}_t \subseteq [t]$ represents the indices of selected data points, i.e., $|\mathcal{I}_T| = m_T$. We define the subspace spanned by the dictionary $\hat{\mathcal{H}}_t = \mathsf{Span}\{\phi(\mathbf{x}_i) : i \in \mathcal{I}_t\} \subseteq \mathcal{H}$ and $\Pi_{\mathcal{D}_t}$ denotes the orthogonal projection onto $\hat{\mathcal{H}}_t$. Specifically, for every $t \in [T-1]$, the above allocation for an observation $\mathbf{x}_{t+1}$ is realized using the following relation:

$$\delta_{t+1} = \left\| \phi(\mathbf{x}_{t+1}) - \Pi_{\mathcal{D}_t} \phi(\mathbf{x}_{t+1}) \right\|_{\mathcal{H}}^2, \tag{3}$$

which admits the kernel form

$$\delta_{t+1} = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}_t^\top(\mathbf{x}_{t+1}) \mathbf{K}_t^{-1} \mathbf{k}_t(\mathbf{x}_{t+1}), \tag{4}$$

where $\mathbf{K}_t$ is the Gram matrix of the dictionary $\mathcal{D}_t$ and $\mathbf{k}_t(\mathbf{x}_{t+1}) = [k(\mathbf{x}_i, \mathbf{x}_{t+1})]_{i \in \mathcal{I}_t}^\top$ is the vector between $\mathbf{x}_{t+1}$ and $\mathcal{D}_t$. If $\delta_{t+1} > \nu$ holds, a new data point $\mathbf{x}_{t+1}$ is added to the dictionary $\mathcal{D}_t$. In this case, the inverse Gram matrix $\mathbf{K}_t^{-1}$ can be updated recursively via the block-matrix inverse:

$$\mathbf{K}_{t+1}^{-1} = \begin{bmatrix} \mathbf{K}_t^{-1} + \delta_{t+1}^{-1} \mathbf{c}_t \mathbf{c}_t^\top & -\delta_{t+1}^{-1} \mathbf{c}_t \\ -\delta_{t+1}^{-1} \mathbf{c}_t^\top & \delta_{t+1}^{-1} \end{bmatrix}, \tag{5}$$

where $\mathbf{c}_t = \mathbf{K}_t^{-1} \mathbf{k}_t(\mathbf{x}_{t+1})$. This formula is derived from the Woodbury identity and allows us to avoid recomputing a matrix inverse from scratch at every step. The dictionary $\mathcal{D}_T$ provides a finite-dimensional feature vector $\psi(\mathbf{x}) = [k(\mathbf{x}_i, \mathbf{x})]_{i \in \mathcal{I}_T}^\top \in \mathbb{R}^m$, where we denote $m \coloneqq m_T$ as the dimension in the feature space for brevity. Importantly, $\psi(\mathbf{x})$ is the vector of inner products between $\phi(\mathbf{x})$ and the dictionary $\mathcal{D}_T$, i.e., $\psi(\mathbf{x}) = [\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle_{\mathcal{H}}]_{i \in \mathcal{I}_T}$. Hence, ψ provides a finite-dimensional representation of ϕ restricted to the subspace $\hat{\mathcal{H}}_T$. Note that this feature vector $\psi(\mathbf{x})$ corresponds to one in Model 1. We conclude this section by introducing the feature matrices as follows:

$$\Psi = [\psi(\mathbf{x}_t)]_{t=1}^T,\ \Psi_0 = [\psi(\mathbf{x}_t)]_{t=1}^{T-1},\ \Psi_1 = [\psi(\mathbf{x}_t)]_{t=2}^T. \tag{6}$$

4.1.2 Regression-based initialization. Here, we explain the coarse parameter estimation based on the Koopman operator regression from the feature matrices. First, we focus on a transition $\mathbf{F} \in \mathbb{R}^{m \times m}$ in the feature space as follows:

$$\min_{\mathbf{F}} \frac{1}{T-1} \|\Psi_1 - \mathbf{F}\Psi_0\|_{\mathrm{F}}^2 + \lambda_A \|\mathbf{F}\|_{\mathrm{F}}^2, \tag{7}$$

where λ_A is a regularization parameter. Reduced rank regression (RRR) [26] interprets the system dynamics as a low-rank linear mapping in the feature space, yielding a spectral initialization for the latent linear system. To solve the problem, we first compute the empirical moment matrices

$$\mathbf{S}_{00} = \frac{1}{T-1} \Psi_0 \Psi_0^\top + \lambda_A \mathbf{I}, \quad \mathbf{S}_{10} = \frac{1}{T-1} \Psi_1 \Psi_0^\top. \tag{8}$$

We then define the whitened cross-covariance $\mathbf{M} = \mathbf{S}_{10} \mathbf{S}_{00}^{-1/2}$, and compute its truncated singular value decomposition $\mathbf{M} \approx \mathbf{U}_r \boldsymbol{\Sigma}_r \mathbf{V}_r^\top$, where $\mathbf{U}_r / \mathbf{V}_r$ denotes the leading r left/right singular vectors and $\boldsymbol{\Sigma}_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ is the corresponding singular values. The dimension of the subspace r is automatically determined by [21]. This procedure identifies the r-dimensional subspace in feature space that best explains the one-step temporal cross-covariance.

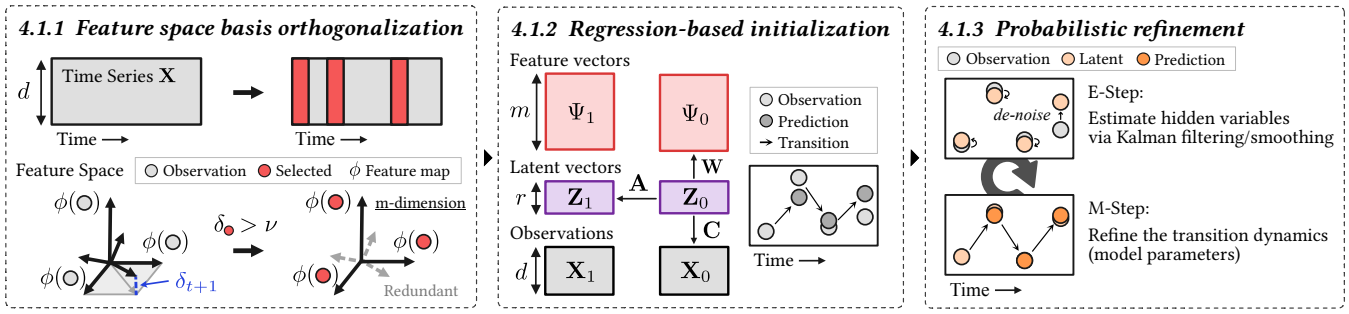


Figure 1: Overview of the static optimization framework. AdaKoop estimates the model parameters θ from a time series X through three stages: (1) *Feature space basis orthogonalization* constructs a sparse dictionary by selecting observations that exceed a linear dependence threshold ν; (2) *Regression-based initialization* performs spectral decomposition on the feature matrices Ψ to initialize the latent linear dynamics $(\tilde{A}, \tilde{C}, \tilde{W})$; and (3) *Probabilistic refinement* iteratively optimizes the latent states via the expectation-maximization (EM) algorithm, leading to dynamical consistency and robustness to noise.

Specifically, we set the projection matrix $\tilde{\mathbf{W}} = \mathbf{U}_r$ and the latent transition matrix $\tilde{\mathbf{A}} = \Sigma_r \mathbf{V}_r^\top \mathbf{S}_{00}^{-1/2} \mathbf{U}_r$. Here, $\tilde{\mathbf{W}}$ spans the dominant predictive feature subspace. Equivalently, when $\tilde{\mathbf{W}}$ has orthonormal columns, $\tilde{\mathbf{A}}$ can be read as the projected dynamics $\tilde{\mathbf{A}} = \tilde{\mathbf{W}}^\dagger \mathbf{F} \tilde{\mathbf{W}}$, where † denotes the Moore–Penrose pseudoinverse.

Next, we estimate the observation matrix $\tilde{\mathbf{C}}$ that maps the latent states back to the observation space. Specifically, we first compute the projected latent states as $\mathbf{Z} = \tilde{\mathbf{W}}^\dagger \Psi$. Then, $\tilde{\mathbf{C}}$ is obtained by a simple linear regression problem, i.e., $\min_{\mathbf{C}} ||\mathbf{X} - \mathbf{C}\mathbf{Z}||_{\mathrm{F}}^2$, yielding the closed-form estimator $\tilde{\mathbf{C}} = \mathbf{X}\mathbf{Z}^\top (\mathbf{Z}\mathbf{Z}^\top)^{-1}$. We omit regularization, such as kernel ridge regression, because the low-dimensional latent space already serves as a regularizer. This concludes the spectral initialization of the latent linear system $\{\tilde{\mathbf{A}}, \tilde{\mathbf{C}}, \tilde{\mathbf{W}}\}$, helping to avoid poor local optima and accelerate convergence.

4.1.3 Probabilistic refinement. Since the regression-based initialization is fitted to one-step feature transitions, the resulting coarse estimators $\{\tilde{\mathbf{A}}, \tilde{\mathbf{C}}, \tilde{\mathbf{W}}\}$ can be sensitive to noise and may be suboptimal in terms of the dynamical consistency of the latent trajectory. To mitigate these issues, we employ the expectation-maximization (EM) algorithm [12] for the refinement of the coarse model parameters and latent vectors. Specifically, we initialize the EM algorithm with the coarse estimators $\{\tilde{\mathbf{A}}, \tilde{\mathbf{C}}, \tilde{\mathbf{W}}\}$. Thanks to the linear representation based on Koopman operator theory, we can avoid computationally expensive sampling-based inference (e.g., particle filtering), commonly required by conventional nonlinear state space models. First, we aim to infer the posterior distributions of the latent vectors $\mathbf{z}_t$. We can efficiently solve these inference problems using the classical sum-product message passing algorithms [48] in probabilistic graphical models. In the context of the linear dynamical system, this inference procedure corresponds to applying the Kalman filter in the forward direction and the Rauch–Tung–Striebel (RTS) smoother in the backward pass. Specifically, the forward passing equations are described by

$$\boldsymbol{\mu}_{t|t-1} = \mathbf{A}\boldsymbol{\mu}_{t-1|t-1} \tag{9}$$

$$\mathbf{P}_{t|t-1} = \mathbf{A}\mathbf{P}_{t-1|t-1}\mathbf{A}^\top + \mathbf{Q} \tag{10}$$

$$\mathbf{G}_t = \mathbf{P}_{t|t-1}\mathbf{H}^\top (\mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^\top + \mathbf{R})^{-1} \tag{11}$$

$$\boldsymbol{\mu}_{t|t} = \boldsymbol{\mu}_{t|t-1} + \mathbf{G}_t(\mathbf{y}_t - \mathbf{H}\boldsymbol{\mu}_{t|t-1}) \tag{12}$$

$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{G}_t\mathbf{H})\mathbf{P}_{t|t-1}, \tag{13}$$

where $\mathbf{G}_t$ is the Kalman gain, $p(\mathbf{z}_t \mid \mathbf{y}_{1:t}) = \mathcal{N}(\boldsymbol{\mu}_{t|t}, \mathbf{P}_{t|t})$ with $\boldsymbol{\mu}_{1|0} = \boldsymbol{\mu}_0$ and $\mathbf{P}_{1|0} = \mathbf{P}_0$. Also, the backward passing equations are

$$\mathbf{J}_t = \mathbf{P}_{t|t}\mathbf{A}^\top \mathbf{P}_{t+1|t}^{-1} \tag{14}$$

$$\boldsymbol{\mu}_{t|T} = \boldsymbol{\mu}_{t|t} + \mathbf{J}_t(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{t+1|t}) \tag{15}$$

$$\mathbf{P}_{t|T} = \mathbf{P}_{t|t} + \mathbf{J}_t(\mathbf{P}_{t+1|T} - \mathbf{P}_{t+1|t})\mathbf{J}_t^\top, \tag{16}$$

where $\mathbf{J}_t$ is the smoother gain and $p(\mathbf{z}_t \mid \mathbf{y}_{1:T}) = \mathcal{N}(\boldsymbol{\mu}_{t|T}, \mathbf{P}_{t|T})$. Based on the smoothed latent vectors $\{\mathbf{z}_t\}$, we update the parameters in closed form:

$$\boldsymbol{\mu}_0^{new} = \boldsymbol{\mu}_{1|T}, \quad \mathbf{P}_0^{new} = \mathbf{P}_{1|T}, \tag{17}$$

$$\mathbf{A}^{new} = \left(\sum_{t=1}^{T-1} \mathbb{E}\left[\mathbf{z}_{t+1}\mathbf{z}_t^\top\right]\right)\left(\sum_{t=1}^{T-1} \mathbb{E}\left[\mathbf{z}_t\mathbf{z}_t^\top\right] + (T-1)\lambda_A \mathbf{I}\right)^{-1}, \tag{18}$$

$$\mathbf{H}^{new} = \left(\sum_{t=1}^{T} \mathbf{y}_t \mathbb{E}\left[\mathbf{z}_t^\top\right]\right)\left(\sum_{t=1}^{T} \mathbb{E}\left[\mathbf{z}_t\mathbf{z}_t^\top\right]\right)^{-1}, \tag{19}$$

and we split $\mathbf{H}$ into $\mathbf{C}$ and $\mathbf{W}$ accordingly. The noise covariances are updated by the residual second moments:

$$\mathbf{Q}^{new} = \frac{1}{T-1}\sum_{t=1}^{T-1} \mathbb{E}\left[\left(\mathbf{z}_{t+1} - \mathbf{A}^{new}\mathbf{z}_t\right)\left(\mathbf{z}_{t+1} - \mathbf{A}^{new}\mathbf{z}_t\right)^\top\right], \tag{20}$$

$$\mathbf{R}_x^{new} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\left(\mathbf{x}_t - \mathbf{C}^{new}\mathbf{z}_t\right)\left(\mathbf{x}_t - \mathbf{C}^{new}\mathbf{z}_t\right)^\top\right], \tag{21}$$

$$\mathbf{R}_\psi^{new} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\left(\psi(\mathbf{x}_t) - \mathbf{W}^{new}\mathbf{z}_t\right)\left(\psi(\mathbf{x}_t) - \mathbf{W}^{new}\mathbf{z}_t\right)^\top\right], \tag{22}$$

where $\mathbb{E}[\mathbf{z}_t] = \boldsymbol{\mu}_{t|T}$, $\mathbb{E}\left[\mathbf{z}_{t+1}\mathbf{z}_t^\top\right] = \mathbf{P}_{t+1|T}\mathbf{J}_t^\top + \boldsymbol{\mu}_{t+1|T}\boldsymbol{\mu}_{t|T}^\top$, and $\mathbb{E}\left[\mathbf{z}_t\mathbf{z}_t^\top\right] = \mathbf{P}_{t|T} + \boldsymbol{\mu}_{t|T}\boldsymbol{\mu}_{t|T}^\top$. Lastly, we introduce the key concept required for the subsequent section and present the theoretical evaluation of time complexity.

Definition 3 (Sufficient statistics set). *Let $\mathcal{S} = \{\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3\}$ be the expected sufficient statistics computed with respect to the*

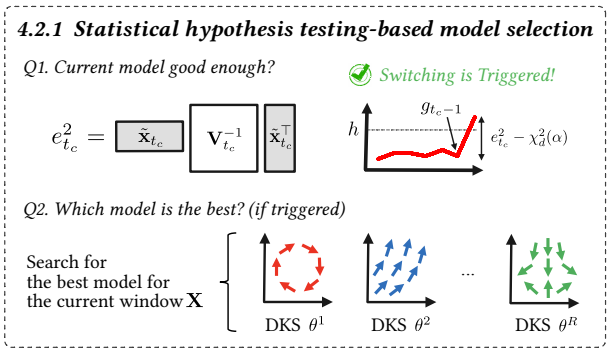


Figure 2: Statistical hypothesis testing-based model selection in AdaKoop: Given new data $\mathbf{x}_{t_c}$, the normalized innovation squared (NIS) is monitored by a one-sided CUSUM test. If the test triggers switching, it evaluates stored models on the current window $\mathbf{X}_c$. When no model handle the current pattern, it estimates new model and adds it to Θ.

smoothed posterior distributions, defined as

$$\mathrm{S}_1 = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\mathbf{z}_t\mathbf{z}_t^\top\right],\quad \mathrm{S}_2 = \frac{1}{T-1}\sum_{t=1}^{T-1}\mathbb{E}\left[\mathbf{z}_{t+1}\mathbf{z}_t^\top\right],\quad \mathrm{S}_3 = \frac{1}{T}\sum_{t=1}^{T}\mathrm{y}_t\mathbb{E}\left[\mathbf{z}_t^\top\right],$$

which are required for the adaptive parameter update in Section 4.2.2.

Lemma 1 (Time complexity of static optimization). *Let m be the final dictionary size produced by the basis orthogonalization, and let r be the latent dimension. Then, the time complexity of the static optimization is $O(Tm^2 + m^3 + \#iter \cdot Tr^2(d+m))$. See Appendix D.1 for details.*

4.2 Streaming Algorithm

We describe how the AdaKoop model is utilized for data stream processing. Algorithm 1 (see Appendix C) describes our streaming algorithm in detail. Given a new data $\mathbf{x}_{t_c}$, our streaming algorithm sequentially applies the following three steps:

- *Statistical hypothesis testing-based model selection*: We employ a change detection mechanism based on the cumulative sum (CUSUM) test [47] of innovation statistics. This step determines the best model for the current window $\mathbf{X}_c$.
- *Adaptive parameter update*: Once a model is selected, we recursively estimate its parameters via an online EM algorithm using sufficient statistics.
- *Forecasting via latent dynamics*: Utilizing the refined parameters and the current latent state estimated by the Kalman filter, we project the system dynamics forward to generate multi-step predictions.

Here, we define the parameter set required for these procedures.

Definition 4 (Model candidate). *Let C be a parameter set for the current window $\mathbf{X}_c = \mathbf{X}[t_s : t_c]$ spanning from the time point t_s to t_c, i.e., $C = \{\boldsymbol{\theta}_c, \mathcal{S}_c, \mathcal{D}_c, \boldsymbol{\mu}_c, \mathbf{P}_c\}$, where $\boldsymbol{\theta}_c$ best summarizes the current window $\mathbf{X}_c$ and $\{\boldsymbol{\mu}_c, \mathbf{P}_c\}$ represents the filtered estimators of the latent state at the current time t_c, which are needed to generate an ℓ_s-steps-ahead future value $\hat{\mathbf{x}}_{t_c+\ell_s}$.*

4.2.1 Statistical hypothesis testing-based model selection. We incrementally update the full parameter set Θ and the model candidate C given a new observation $\mathbf{x}_{t_c}$ and the current window $\mathbf{X}_c = \mathbf{X}[t_s : t_c]$. We monitor the normalized innovation squared (NIS) derived from the Kalman filter innovations, which follows a χ^2 law under the linear Gaussian assumption. The one-step-ahead prediction is

$$\hat{\boldsymbol{\mu}}_c^{new} = \mathbf{A}\boldsymbol{\mu}_c^{prev}, \quad \hat{\mathbf{P}}_c^{new} = \mathbf{A}\mathbf{P}_c^{prev}\mathbf{A}^\top + \mathbf{Q}. \tag{23}$$

The corresponding innovation for the observed channel is

$$\tilde{\mathbf{x}}_{t_c} = \mathbf{x}_{t_c} - \mathbf{C}\hat{\boldsymbol{\mu}}_c^{new}, \quad \mathbf{V}_c = \mathbf{C}\hat{\mathbf{P}}_c^{new}\mathbf{C}^\top + \mathbf{R}_x, \tag{24}$$

and we define the normalized innovation squared (NIS) by

$$e_{t_c}^2 = \tilde{\mathbf{x}}_{t_c}^\top \mathbf{V}_c^{-1}\tilde{\mathbf{x}}_{t_c}. \tag{25}$$

Under the nominal model, $e_{t_c}^2$ approximately follows a χ^2 distribution with d degrees of freedom. Thus, given a confidence level $\alpha \in (0,1)$, we set a threshold $\chi_d^2(1-\alpha)$ and apply a one-sided CUSUM test:

$$g_{t_c} = \max\left\{0,\ g_{t_c-1} + \left(e_{t_c}^2 - \chi_d^2(1-\alpha)\right)\right\}. \tag{26}$$

If g_{t_c} exceeds a switching limit h, we consider the current model to be inconsistent and search for a better model.

When switching is triggered, it validates each model $\boldsymbol{\theta} \in \Theta$ using the current window $\mathbf{X}_c$. For every candidate $\boldsymbol{\theta} \in \Theta$, we reset its latent state to a broad prior; then refine the initial latent moments via a smoothing-based initialization. After this initialization, we run a Kalman filter over the window and compute the NIS sequence $\{e_t^2\}_{t=t_s}^{t_c}$ based on Eq. (25). We select the model that minimizes the mean value of the NIS sequence. However, if there is no model such that the maximum value of the NIS sequence is lower than h, AdaKoop creates a new model from $\mathbf{X}_c$ via the static optimization and inserts it into Θ to handle an unknown pattern adaptively.

4.2.2 Adaptive parameter update. Once a model $\boldsymbol{\theta}_c$ is selected, it performs a one-step online EM update with a forgetting factor $\gamma \in (0,1]$. This update consists of (i) feature-space adaptation by Eq. (3) and (ii) recursive updates of sufficient statistics. Given a new point $\mathbf{x}_{t_c}$, we compute the residual δ_{t_c} as in Eq. (3). First, if $\delta_{t_c} > \nu$ holds, we add $\mathbf{x}_{t_c}$ to $\mathcal{D}_c$ and obtain the coefficient vector $\mathbf{a}_{t_c} = \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}_{t_c})$. When the dictionary grows from m to $m+1$, the augmented observation map $\mathbf{H}$ must be expanded accordingly. We expand $\mathbf{W}$ by one row via $\mathbf{a}_{t_c}^\top\mathbf{W}$ and extend the feature-noise covariance $\mathbf{R}_\psi$ by a block-diagonal update. Similarly, we expand the sufficient statistic S_3 so that its dimension matches $d+m$. This step prevents reinitialization and enables stable updates even when the feature space is growing. If the dictionary size exceeds $m_{\max}$ after an update, we employ a pruning strategy to remove the basis vector that is most linearly dependent on the others, thereby minimizing the loss of information in the feature space. Specifically, it is a known property of the Schur complement that the reciprocal of the j-th diagonal element $1/[\mathbf{K}^{-1}]_{jj}$ equals the squared residual error of approximating the j-th basis vector using the linear span of the remaining $m-1$ vectors. Therefore, we select the index $j^* = \arg\max_j [\mathbf{K}^{-1}]_{jj}$ for removal. We then update $\mathbf{K}^{-1}$ by applying a rank-1 downdate operation, which is mathematically equivalent to reversing the recursive update in Eq. (5), and delete the corresponding rows and columns from $\mathbf{W}$, $\mathbf{R}_\psi$, and S_3.

Table 2: Multivariate forecasting results averaged across 71 datasets, where the best results are in bold and the second best are underlined. AdaKoop consistently outperforms its baselines. Full results for all datasets are provided in Appendix E.2.

Metrics	*Mean Squared Error (MSE)*			*Mean Absolute Error (MAE)*		
Steps	20	25	30	20	25	30
sKAF (2023)	34.0 ± 112	59.6 ± 285	81.4 ± 270	1.20 ± 1.96	1.41 ± 2.35	1.72 ± 2.61
Koopa (2023)	0.397 ± 0.417	0.377 ± 0.331	0.361 ± 0.338	0.430 ± 0.165	0.427 ± 0.154	0.423 ± 0.144
OneNet (2023)	0.318 ± 0.148	0.321 ± 0.146	<u>0.322 ± 0.145</u>	0.441 ± 0.123	0.443 ± 0.122	0.444 ± 0.121
PAttn (2024)	0.311 ± 0.206	0.326 ± 0.206	0.330 ± 0.196	0.407 ± 0.139	0.417 ± 0.138	0.421 ± 0.134
WPMixer (2025)	0.561 ± 1.95	0.509 ± 1.56	0.436 ± 0.916	0.411 ± 0.233	<u>0.413 ± 0.214</u>	<u>0.413 ± 0.194</u>
ModePlait (2025)	<u>0.279 ± 0.0918</u>	<u>0.300 ± 0.104</u>	0.325 ± 0.115	<u>0.407 ± 0.0857</u>	0.423 ± 0.0914	0.441 ± 0.0964
AdaKoop (Ours)	**0.0763 ± 0.0539**	**0.0999 ± 0.0659**	**0.120 ± 0.0747**	**0.183 ± 0.0769**	**0.214 ± 0.0857**	**0.240 ± 0.0888**

Next, given $(\hat{\boldsymbol{\mu}}_c^{new}, \hat{\mathbf{P}}_c^{new})$ in Eq. (23), we apply one Kalman update to obtain the filtered state $(\boldsymbol{\mu}_c^{new}, \mathbf{P}_c^{new})$. We introduce the sufficient statistics set rule. Using the filtered moments $\mathbb{E}[\mathbf{z}_{t_c}\mathbf{z}_{t_c}^\top] = \mathbf{P}_c^{new} + \boldsymbol{\mu}_c^{new}(\boldsymbol{\mu}_c^{new})^\top$ and a filtered approximation of $\mathbb{E}[\mathbf{z}_{t_c}\mathbf{z}_{t_c-1}^\top]$, we update

$$\mathbf{S}_1^{new} \leftarrow (1-\gamma)\,\mathbf{S}_1^{prev} + \gamma\,\mathbb{E}[\mathbf{z}_{t_c}\mathbf{z}_{t_c}^\top], \tag{27}$$

$$\mathbf{S}_2^{new} \leftarrow (1-\gamma)\,\mathbf{S}_2^{prev} + \gamma\,\mathbb{E}[\mathbf{z}_{t_c}\mathbf{z}_{t_c-1}^\top], \tag{28}$$

$$\mathbf{S}_3^{new} \leftarrow (1-\gamma)\,\mathbf{S}_3^{prev} + \gamma\,\mathbf{y}_{t_c}\mathbb{E}[\mathbf{z}_{t_c}^\top]. \tag{29}$$

Then, the parameters are updated in closed form by

$$\mathbf{A}^{new} \leftarrow \mathbf{S}_2^{new}(\mathbf{S}_1^{prev} + \lambda_A\mathbf{I})^{-1}, \tag{30}$$

$$\mathbf{H}^{new} \leftarrow \mathbf{S}_3^{new}(\mathbf{S}_1^{new})^{-1}. \tag{31}$$

4.2.3 Forecasting via latent dynamics. Finally, it forecasts future values using the model candidate C. Let $\boldsymbol{\mu}_c^{new}$ be the filtered latent mean after processing $\mathbf{x}_{t_c}$. Because the latent dynamics are linear, the ℓ_s-steps-ahead future value is simply computed by

$$\hat{\mathbf{z}}_{t_c+\ell_s} = \mathbf{A}^{\ell_s}\boldsymbol{\mu}_c^{new}, \quad \hat{\mathbf{x}}_{t_c+\ell_s} = \mathbf{C}\hat{\mathbf{z}}_{t_c+\ell_s}. \tag{32}$$

This forecasting step is computationally light and directly leverages the current model selected by the statistical tests.

Lemma 2 (Time complexity of AdaKoop). *Let m be the dictionary size, R be the number of stored models, and r be the latent dimension. The time complexity of AdaKoop is at least $O(m^2 + r^2(d+m))$ and at most $O(Tm^2 + m^3 + RTr^2(d+m) + \#iter \cdot Tr^2(d+m))$ per process. See Appendix D.2 for details.*

Lemma 2 indicates that the computational time of AdaKoop is constant with regard to the length of a data stream t_c. Thus, it is practical for semi-infinite data streams in terms of execution speed.

5 Experiments

We ran experiments to answer the following questions.

- *Q1. Accuracy*: How accurately does it forecast future values?
- *Q2. Scalability*: How does it scale in computational time?
- *Q3. Nonlinear capability*: How well does it estimate and leverage complex nonlinear dynamics in a streaming fashion?
- *Q4. Sensitivity analysis*: How do hyperparameters influence real-time forecasting performance?

5.1 Experimental Setup

5.1.1 Datasets. We used the dysts benchmark [23], which offers a diverse and practical collection of 71 chaotic dynamical systems for evaluating time series forecasting and data-driven modeling techniques. This benchmark is particularly noteworthy for its wide applicability across diverse fields, including astrophysics, climatology, and biochemistry. Note that chaotic dynamical systems are a kind of nonlinear dynamical system, and some of them consist of multiple distinct dynamic patterns [16]. Hence, this benchmark is suitable for our experimental evaluation.

5.1.2 Baselines. We compared AdaKoop with the following seven models for time series forecasting.

- ModePlait [11] is a streaming algorithm for the discovery of time-evolving causality and forecasting future values.
- WPMixer [45] is an MLP-based method that uses multi-level wavelet decomposition to integrate information from both time and frequency domains.
- PAttn [55] is a simple linear model with patching and attention for feature extraction.
- OneNet [56] is an online ensembling network that handles concept drift by utilizing both channel independence and cross-variable dependency.
- Koopa [38] disentangles non-stationary time series into time-variant and time-invariant components by approximating the Koopman operator.
- Streaming KAF (sKAF) [22] reduces computational costs by approximation using randomized features and Nyström methods.

5.2 Results

5.2.1 Q1. Accuracy. We first assessed the quality of AdaKoop in terms of ℓ_s-steps-ahead forecasting accuracy. We used two evaluation metrics, the mean square error (MSE) and the mean absolute error (MAE), both of which provide better results when they are closer to zero. Table 2 shows the mean and standard deviation of the results across 5 different seed values and 71 datasets, where the best and second-best levels of performance are shown in **bold** and <u>underlined</u>, respectively. We provide the critical difference diagrams of these results and the full results for all datasets in Appendix E.2. AdaKoop outperforms state-of-the-art baselines by employing the Koopman operator for the estimation of nonlinear dynamics and the adaptive processing of nonstationary data streams. Although deep

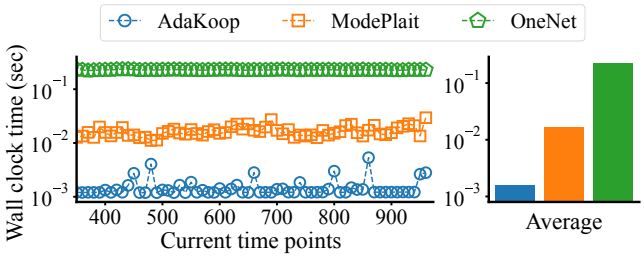


Figure 3: Scalability of AdaKoop: (left) Wall clock time vs. data stream length t_c and (right) average time consumption.

Table 3: Forecasting results across different kernel functions.

Metrics		MSE	MAE
RBF	20	0.0763 ± 0.0539	0.183 ± 0.0769
	25	0.0999 ± 0.0659	0.214 ± 0.0857
	30	0.120 ± 0.0747	0.240 ± 0.0888
Polynomial	20	0.115 ± 0.0922	0.221 ± 0.0953
	25	0.144 ± 0.108	0.254 ± 0.102
	30	0.164 ± 0.114	0.278 ± 0.103
Sigmoid	20	0.218 ± 0.0868	0.365 ± 0.0837
	25	0.239 ± 0.0967	0.384 ± 0.0889
	30	0.253 ± 0.101	0.398 ± 0.0920
Linear	20	0.218 ± 0.0870	0.366 ± 0.0846
	25	0.239 ± 0.0963	0.386 ± 0.0893
	30	0.253 ± 0.0999	0.400 ± 0.0917

learning-based models exhibit high generality for time series modeling, they cannot adapt to time-varying environments and thus degrade significantly on datasets with distinct dynamic patterns. ModePlait summarizes an entire data stream through the discovery of time-evolving causality. However, since it is not specialized for modeling nonlinear dynamical systems, its performance may degrade for highly complex phenomena. Streaming KAF showed considerable instability in forecasting accuracy because its update rule cannot handle unknown dynamic patterns. The results also showed that methods capable of handling time-varying dynamic patterns exhibit relatively small standard deviations.

5.2.2 Q2. Scalability. We evaluated the performance of AdaKoop in terms of computational time. Figure 3 compares the computational efficiency of methods that handle time-varying nonlinear dynamic patterns in a data stream. It presents the computational time at each time point t_c on the left and the average time on the right. Note that both figures are shown on linear-log scales. AdaKoop consistently outperformed its competitors in terms of computational efficiency, thanks to the efficient streaming algorithm. In addition, this result aligns with the discussion provided in Lemma 2. OneNet requires significantly more time for model updates than other methods due to the use of two complementary models. Although ModePlait operates at a moderate speed, it not only forecasts but also discovers causal relationships sequentially, which makes it slower than our algorithm.

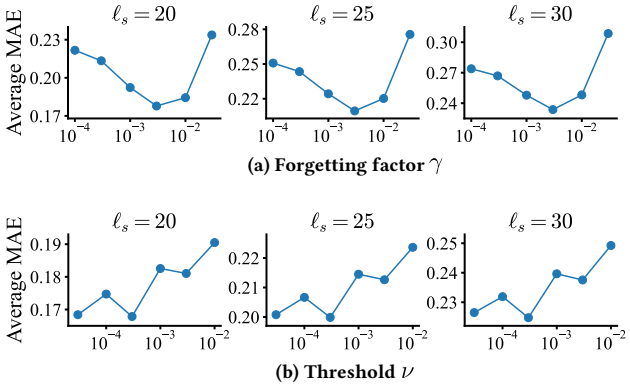


Figure 4: Sensitivity analysis results.

5.2.3 Q3. Nonlinear capability. We demonstrated that AdaKoop accurately captures nonlinear dynamics and utilizes the learned models for real-time forecasting. To validate this, we compared the RBF kernel, which meets our theoretical requirements, with other suboptimal kernels. Table 3 shows the forecasting results across various kernel functions. The RBF kernel consistently yields the best forecasting performance, demonstrating its superior capability in capturing complex nonlinear dynamics as mentioned in Section 5.2.1. While the polynomial kernel achieves relatively high performance, it falls short of the RBF kernel because this kernel maps data into a finite-dimensional space. In contrast, the linear kernel fails to capture complex nonlinear dynamics, leading to a substantial decrease in forecasting accuracy. Although the sigmoid kernel is an infinite-dimensional kernel, it is not positive definite. Consequently, it violates the theoretical requirements of the Koopman operator regression framework, resulting in performance comparable to that of a linear kernel. These results empirically demonstrate that our probabilistic augmented state space model effectively leverages RKHS properties to estimate nonlinear dynamics despite being in linear system form and accurately performs real-time forecasting in a nonstationary environment.

5.2.4 Q4. Sensitivity analysis. We evaluated the sensitivity of our algorithm to its hyperparameters. Figure 4 shows the forecasting accuracy (MAE) when varying hyperparameters. Figure 4 (a) shows the impact of the forgetting factor γ. As γ increases, AdaKoop incorporates more recent information and discards the past one. This result indicates that the best γ lies in $[10^{-3}, 10^{-2}]$. When γ is too small, the algorithm adapts too slowly to changes in a nonstationary data stream, resulting in poor tracking performance. Conversely, a large γ results in unstable estimation due to the insufficient effective sample size. Next, Figure 4 (b) illustrates the sensitivity to the threshold ν. This parameter controls the sparsity of the dictionary in the feature space. We can observe that a tendency towards improved accuracy as ν decreases. This is because that a smaller threshold enables the model to maintain a richer dictionary of basis vectors, allowing for a more precise approximation of complex nonlinear dynamics. Note that small ν improves model expressiveness but increases computational costs, so we should select an appropriate value that balances accuracy and efficiency according to the requirements.

6 Conclusion

In this paper, we presented an efficient streaming method, AdaKoop, designed for modeling nonlinear dynamics from nonstationary data streams with Koopman operator regression. Our leading solution relies on a probabilistic dual-view state space model that treats both raw observations and RKHS features as emissions from a shared latent linear system, thereby enabling stable and efficient modeling of nonlinear dynamics. To adaptively handle nonstationarity in data streams, our algorithm employs statistical hypothesis testing and continuous parameter updates, but its computational complexity remains constant regardless of the data stream length. Experimental results across 71 practical benchmark datasets showed that AdaKoop achieved remarkable improvements over competitors in both forecasting accuracy and computational efficiency. The results provide empirical validation that AdaKoop is suitable for real-time forecasting in nonstationary data streams.

Acknowledgments

We would like to thank the anonymous referees for their valuable and helpful comments. This work was partly supported by "Program for Leading Graduate Schools" of the Osaka University, Japan, JSPS KAKENHI Grant-in-Aid for Scientific Research Number JP25KJ1729, JP26H02499, JST CREST JPMJCR23M3, JST K Program JPMJKP25Y6, JST COI-NEXT JPMJPF2009, JST COI-NEXT JPMJPF2115, Future Social Value Co-Creation Project - Osaka University.

References

[1] Charu C. Aggarwal (Ed.). 2007. *Data Streams - Models and Algorithms*. Advances in Database Systems, Vol. 31. Springer.
[2] Nachman Aronszajn. 1950. Theory of reproducing kernels. *Transactions of the American mathematical society* 68, 3 (1950), 337–404.
[3] Nimrod Berman, Ilan Naiman, and Omri Azencot. 2023. Multifactor Sequential Disentanglement via Structured Koopman Autoencoders. In *Proceedings of the 11th International Conference on Learning Representations*.
[4] Dimitris Bertsimas and Wes Gurnee. 2023. Learning sparse nonlinear dynamics via mixed-integer optimization. *Nonlinear Dynamics* 111, 7 (2023), 6585–6604.
[5] George EP Box and Gwilym M Jenkins. 1976. *Time series analysis: forecasting and control (Revised Edition)*. John Wiley & Sons.
[6] Steven L. Brunton, Marko Budisic, Eurika Kaiser, and J. Nathan Kutz. 2022. Modern Koopman Theory for Dynamical Systems. *SIAM Rev.* 64, 2 (2022), 229–340.
[7] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences* 113, 15 (2016), 3932–3937.
[8] Marko Budišić, Ryan Mohr, and Igor Mezić. 2012. Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22, 4 (2012).
[9] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. Brits: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc.
[10] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc.
[11] Naoki Chihara, Yasuko Matsubara, Ren Fujiwara, and Yasushi Sakurai. 2025. Modeling time-evolving causality over data streams. In *Proceedings of the 31st ACM SIGKDD international conference on Knowledge discovery and data mining*. Association for Computing Machinery, 153–164.
[12] A P Dempster, N M Laird, and D B Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society. Series B, Statistical methodology* 39, 1 (1 Sept. 1977), 1–22.
[13] Pramith Devulapalli and Steve Hanneke. 2024. Learning from snapshots of discrete and continuous data streams. *Advances in Neural Information Processing Systems* 37 (2024), 80082–80106.
[14] Gerardo Duran-Martin, Matias Altamirano, Alexander Y Shestopaloff, Leandro Sánchez-Betancourt, Jeremias Knoblauch, Matt Jones, François-Xavier Briol, and Kevin Murphy. 2024. Outlier-robust Kalman Filtering through Generalised Bayes. In *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235. 12138–12171.
[15] Yaakov Engel, Shie Mannor, and Ron Meir. 2004. The kernel recursive least-squares algorithm. *IEEE Transactions on signal processing* 52, 8 (2004), 2275–2285.
[16] Amirreza Farnoosh, Bahar Azari, and Sarah Ostadabbas. 2021. Deep switching auto-regressive factorization: Application to time series forecasting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, Vol. 35. 7394–7403.
[17] L Foresti, M Reyniers, A Seed, and L Delobbe. 2016. Development and verification of a real-time stochastic precipitation nowcasting system for urban hydrology in Belgium. *Hydrology and Earth System Sciences* 20, 1 (29 Jan. 2016), 505–527.
[18] Ren Fujiwara, Yasuko Matsubara, and Yasushi Sakurai. 2025. Modeling Latent Non-Linear Dynamical System over Time Series. In *Proceedings of the 39th AAAI conference on artificial intelligence*.
[19] Ren Fujiwara, Yasuko Matsubara, and Yasushi Sakurai. 2026. When to Retrain after Drift: A Data-Only Test of Post-Drift Data Size Sufficiency. In *The Fourteenth International Conference on Learning Representations*.
[20] João Gama and Guangquan Zhang. 2019. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering* 31, 12 (2019), 2346–2363.
[21] Matan Gavish and David L Donoho. 2014. The Optimal Hard Threshold for Singular Values is <inline-formula> $4/\sqrt{3}$ </inline-formula>. *IEEE Transactions on Information Theory* 60, 8 (Aug. 2014), 5040–5053.
[22] Dimitrios Giannakis, Amelia Henriksen, Joel A Tropp, and Rachel Ward. 2023. Learning to Forecast Dynamical Systems from Streaming Data. *SIAM Journal on Applied Dynamical Systems* 22, 2 (2023), 527–558.
[23] William Gilpin. 2021. Chaos as an interpretable benchmark for forecasting and data-driven modelling. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc.
[24] V Gupta, M Mittal, and V Mittal. 2019. R-peak detection using chaos analysis in standard and real time ECG databases. *IRBM* 40, 6 (Dec. 2019), 341–354.
[25] Shingo Higashiguchi, Yasuko Matsubara, Koki Kawabata, Taichi Murayama, and Yasushi Sakurai. 2025. D-Tracker: Modeling interest diffusion in social activity tensor data streams. In *Proceedings of the 31st ACM SIGKDD international conference on Knowledge discovery and data mining*. Association for Computing Machinery, 460–471.
[26] Alan Julian Izenman. 1975. Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis* 5, 2 (1975), 248–264.
[27] Soshi Kakio, Yasuko Matsubara, Ren Fujiwara, and Yasushi Sakurai. 2026. Multi-Aspect Mining and Anomaly Detection for Heterogeneous Tensor Streams. In *Proceedings of the ACM Web Conference 2026*. Association for Computing Machinery, 6989–7000.
[28] Haik Kalantarian, Costas Sideris, Bobak Mortazavi, Nabil Alshurafa, and Majid Sarrafzadeh. 2017. Dynamic computation offloading for low-power wearable health monitoring systems. *IEEE Transactions on Bio-Medical Engineering* 64, 3 (March 2017), 621–628.
[29] R E Kalman. 1963. Mathematical description of linear dynamical systems. *J. Soc. Indust. Appl. Math.* 1, 2 (Jan. 1963), 152–192.
[30] Yoshinobu Kawahara. 2016. Dynamic mode decomposition with reproducing kernels for Koopman spectral analysis. In *Advances in Neural Information Processing Systems*, Vol. 29. Curran Associates, Inc.
[31] George Kimeldorf and Grace Wahba. 1971. Some results on Tchebycheffian spline functions. *Journal of mathematical analysis and applications* 33, 1 (1971), 82–95.
[32] DP Kingma and J Adam Ba. 2015. A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.
[33] Stefan Klus, Ingmar Schuster, and Krikamol Muandet. 2020. Eigendecompositions of transfer operators in reproducing kernel Hilbert spaces. *Journal of Nonlinear Science* 30 (2020), 283–315.
[34] Bernard O Koopman. 1931. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences* 17, 5 (1931), 315–318.
[35] Milan Korda and Igor Mezić. 2018. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica: The Journal of IFAC, the International Federation of Automatic Control* 93 (July 2018), 149–160.
[36] Vladimir Kostic, Karim Lounici, Pietro Novelli, and Massimiliano Pontil. 2023. Sharp spectral rates for Koopman operator learning. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 32328–32339.
[37] Vladimir Kostic, Pietro Novelli, Andreas Maurer, Carlo Ciliberto, Lorenzo Rosasco, and Massimiliano Pontil. 2022. Learning dynamical systems via Koopman operator regression in reproducing kernel Hilbert spaces. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., 4017–4031.
[38] Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. 2023. Koopa: Learning non-stationary time series dynamics with koopman predictors. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 12271–12290.
[39] Lars Lorch, Andreas Krause, and Bernhard Schölkopf. 2024. Causal Modeling with Stationary Diffusions. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (Eds.), Vol. 238. 1927–1935.
[40] Edward N Lorenz. 1963. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences* 20, 2 (1963), 130–141.

[41] Yasuko Matsubara and Yasushi Sakurai. 2025. MicroAdapt : Self-evolutionary dynamic modeling algorithms for time-evolving data streams. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Vol. 1. Association for Computing Machinery, 2114–2125.

[42] Shane A McQuarrie, Cheng Huang, and Karen E Willcox. 2021. Data-driven reduced-order models via regularised Operator Inference for a single-injector combustion process. *Journal of the Royal Society of New Zealand* 51, 2 (3 April 2021), 194–211.

[43] Giacomo Meanti, Antoine Chatalic, Vladimir Kostic, Pietro Novelli, Massimiliano Pontil, and Lorenzo Rosasco. 2023. Estimating Koopman operators with sketching to provably learn large scale dynamical systems. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 77242–77276.

[44] Igor Mezić. 2005. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics* 41 (2005), 309–325.

[45] Md Mahmuddun Nabi Murad, Mehmet Aktukmak, and Yasin Yilmaz. 2025. Wpmixer: Efficient multi-resolution mixing for long-term time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 19581–19588.

[46] Kota Nakamura, Koki Kawabata, Yasuko Matsubara, and Yasushi Sakurai. 2026. Fast Mining and Dynamic Time-to-Event Prediction over Multi-sensor Data Streams. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 1089–1100.

[47] E S Page. 1954. Continuous Inspection Schemes. *Biometrika* 41, 1/2 (June 1954), 100.

[48] Judea Pearl. 1982. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Probabilistic and causal inference: the works of Judea Pearl*. 133–136.

[49] Xihao Piao, Zheng Chen, Taichi Murayama, Yasuko Matsubara, and Yasushi Sakurai. 2024. Fredformer: Frequency debiased transformer for time series forecasting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2400–2410.

[50] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. 2016. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems* 15, 1 (2016), 142–161.

[51] Michael Roth, Emre Ozkan, and Fredrik Gustafsson. 2013. A Student's t filter for heavy tailed process and measurement noise. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 5770–5774.

[52] Peter J Schmid. 2010. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics* 656 (2010), 5–28.

[53] Naoya Takeishi, Yoshinobu Kawahara, Yasuo Tabei, and Takehisa Yairi. 2017. Bayesian dynamic mode decomposition.. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2814–2821.

[54] Naoya Takeishi, Takehisa Yairi, and Yoshinobu Kawahara. 2018. Factorially switching dynamic mode decomposition for Koopman analysis of time-variant systems. In *IEEE Conference on Decision and Control*. 6402–6408.

[55] Mingtian Tan, Mike A Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. 2024. Are Language Models Actually Useful for Time Series Forecasting?. In *Advances in Neural Information Processing Systems*, Vol. 37. Curran Associates, Inc., 60162–60191.

[56] Qingsong Wen, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, Tieniu Tan, et al. 2023. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. In *Advances in Neural Information Processing Systems*, Vol. 36. Curran Associates, Inc., 69949–69980.

[57] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* 1, 6 (Dec. 1945), 80–83.

[58] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. 2015. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science* 25 (2015), 1307–1346.

[59] Hao Zhang, Clarence W Rowley, Eric A Deem, and Louis N Cattafesta. 2019. Online dynamic mode decomposition for time-varying systems. *SIAM Journal on Applied Dynamical Systems* 18, 3 (2019), 1586–1609.

[60] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the 35th AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.

Appendix

A Notation and Terminology

Table 4 lists the symbols and definitions used in our work.

Table 4: Symbols and definitions.

Symbol	Definition
General notation	
d	Number of dimensions
T	Window size
t_c	Current time point
$\mathbf{x}_t$	Observation at time t
$\mathbf{X}$	Nonstationary multivariate data stream (semi-infinite)
$\mathbf{X}_c$	Current window, i.e., $\mathbf{X}_c = \mathbf{X}[t_s : t_c] \in \mathbb{R}^{d \times T}$
$\mathcal{H}$	Reproducing kernel Hilbert space (RKHS)
ϕ	Feature map, i.e., $\phi : \mathcal{M} \to \mathcal{H}$
k	Kernel function, i.e., $k = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathcal{H}}$
$\dagger$	Moore–Penrose pseudoinverse
Dual-view kernelized system (DKS)	
m	Dimension of feature space
r	Dimension of latent space
$\mathbf{y}_t$	Augmented observations (two-views)
$\mathbf{z}_t$	Latent state vector
$\boldsymbol{\theta}$	DKS model, i.e., $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{Q}, \mathbf{C}, \mathbf{W}, \mathbf{R}_x, \mathbf{R}_\psi, \boldsymbol{\mu}_0, \mathbf{P}_0\}$
R	Number of models
$\boldsymbol{\Theta}$	DKS model set, i.e., $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i=1}^{R}$
Static optimization	
$\mathcal{I}_t$	Indices of selected data points
$\mathcal{D}_t$	Dictionary, i.e., $\mathcal{D} = \{\mathbf{x}_\tau \mid \tau \in \mathcal{I}_t\}$
$\hat{\mathcal{H}}_t$	Dictionary subspace, i.e., $\hat{\mathcal{H}}_t = \mathsf{Span}\{\phi(\mathbf{x}_i) : i \in \mathcal{I}_t\} \subseteq \mathcal{H}$
$\Pi_{\mathcal{D}_t}$	Orthogonal projection onto $\hat{\mathcal{H}}$
$\mathbf{K}_t$	Gram matrix over $\mathcal{D}_t$
$\mathbf{k}_t(\mathbf{x})$	Kernel vector between $\mathbf{x}$ and $\mathcal{D}_t$
δ_{t+1}	Residual distance of $\phi(\mathbf{x}_{t+1})$ to $\hat{\mathcal{H}}_t$
$\psi(\mathbf{x})$	Feature vector based on dictionary, i.e., $\psi(\mathbf{x}) = [k(\mathbf{x}_i, \mathbf{x})]_{i \in \mathcal{I}_T}^\top$
Ψ	Feature matrix, i.e., $\Psi = [\psi(\mathbf{x}_t)]_{t=1}^{T}$
λ_A	Regularization parameter
$\{\tilde{\mathbf{A}}, \tilde{\mathbf{C}}, \tilde{\mathbf{W}}\}$	Coarse estimators for initializing the EM algorithm
$\mathbf{G}_t$	Kalman gain
$\mathbf{J}_t$	Smoother gain
$\#iter$	Number of EM iterations
Streaming algorithm	
$\mathcal{S}$	Sufficient statistical set, i.e., $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3\}$
γ	Forgetting factor
ℓ_s	Forecasting steps
$\boldsymbol{\mu}_c, \mathbf{P}_c$	Filtered latent mean/covariance at time t_c
C	Model candidate, i.e., $C = \{\theta_c, \mathcal{S}_c, \mathcal{D}_c, \boldsymbol{\mu}_c, \mathbf{P}_c\}$
$\tilde{\mathbf{x}}_{t_c}$	Innovation, i.e., $\tilde{\mathbf{x}}_{t_c} = \mathbf{x}_{t_c} - \mathbf{C}\hat{\boldsymbol{\mu}}_c^{new}$
$e_{t_c}^2$	Normalized innovation squared (NIS)
α	Confidence level
g_{t_c}	One-sided CUSUM statistic computed from the NIS sequence

B Preliminaries

B.1 Koopman Operator Theory

Let $X \coloneqq \{X_t : t \in \mathbb{N}\}$ be a family of random variables with values in a measurable space $(\mathcal{M}, \Sigma_{\mathcal{M}})$, called state space. We assume that X is a time-homogeneous[1] Markov chain[2] with a transition kernel $p : \mathcal{M} \times \Sigma_{\mathcal{M}} \to [0, 1]$ such that

$$P(X_{t+1} \in B \mid X_t = x) = p(x, B), \quad \forall t \in \mathbb{N},\ \forall (x, B) \in \mathcal{M} \times \Sigma_{\mathcal{M}}.$$

For a vector space $\mathcal{F}$ of $\Sigma_{\mathcal{M}}$-measurable observables $\eta : \mathcal{M} \to \mathbb{C}$, the Markov transfer operator $\mathcal{K}_{\mathcal{F}} : \mathcal{F} \to \mathcal{F}$ is defined by

$$(\mathcal{K}_{\mathcal{F}}\eta)(x) \coloneqq \int_{\mathcal{M}} \eta(y) p(x, dy) = \mathbb{E}[\eta(X_{t+1}) \mid X_t = x],$$

[1] Time-homogeneity denotes that $P(X_{t+1} \in B \mid X_t = x)$ for any measurable set B does not depend on t.

[2] Markov chain denotes that X_{t+1} only depends on the present state X_t.

Algorithm 1 AdaKoop ($\mathbf{X}_c, \Theta, C$)

Input: (a) Current window $\mathbf{X}_c \in \mathbb{R}^{d\times T}$ at current time t_c
(b) Full parameter set Θ,
(c) Model candidate $C = \{\boldsymbol{\theta}_c, \mathcal{S}_c, \mathcal{D}_c, \boldsymbol{\mu}_c, \mathbf{P}_c\}$
Output: (a) Updated full parameter set Θ'
(b) Updated model candidate C'
(c) ℓ_s-steps-ahead future value $\hat{\mathbf{x}}_{t_c+\ell_s}$

$\hat{\boldsymbol{\mu}}_c = \mathbf{A}\boldsymbol{\mu}_c$, $\hat{\mathbf{P}}_c = \mathbf{A}\mathbf{P}_c\mathbf{A}^\top + \mathbf{Q}$ ▷ Eq. (23)
$\tilde{\mathbf{x}}_{t_c} = \mathbf{x}_{t_c} - \mathbf{C}\hat{\boldsymbol{\mu}}_c$, $\mathbf{V}_c = \mathbf{C}\hat{\mathbf{P}}_c\mathbf{C}^\top + \mathbf{R}_x$ ▷ Eq. (24)
$e^2_{t_c} = \tilde{\mathbf{x}}_{t_c}^\top \mathbf{V}_c^{-1}\tilde{\mathbf{x}}_{t_c}$ ▷ Eq. (25)
$g_{t_c} = \max\left\{0,\ g_{t_c-1} + \left(e^2_{t_c} - \chi^2_d(1-\alpha)\right)\right\}$ ▷ Eq. (26)
if $g_{t_c} > h$ **then**
  $\Theta_h \leftarrow \{\boldsymbol{\theta} \in \Theta : \max(\{e^2_t\}_{t=t_s}^{t_c}) < h\}$
  if $|\Theta_h| = 0$ **then**
    $C' \leftarrow$ StaticOptimization($\mathbf{X}_c$) ▷ Algorithm 2
    $\Theta' \leftarrow \Theta \cup \boldsymbol{\theta}_c$
  else
    $\boldsymbol{\theta}_c \leftarrow \arg\min_{\boldsymbol{\theta}\in\Theta_h} \text{mean}(\{e^2_t\}_{t=t_s}^{t_c})$
  end if
end if
if NOT create new model **then**
  Update dictionary $\mathcal{D}'_c$ ▷ Eqs. (3)-(5)
  Update filtered estimators $\boldsymbol{\mu}'_c, \mathbf{P}'_c$ ▷ Eqs. (9)-(13)
  Update $\mathcal{S}'_c$, $\mathbf{A}'$, $\mathbf{H}'$ ▷ Eqs. (27)-(31)
  $C' \leftarrow \{\boldsymbol{\theta}_c, \mathcal{D}'_c, \mathcal{S}'_c, \boldsymbol{\mu}'_c, \mathbf{P}'_c\}$
end if
$\hat{\mathbf{x}}_{t_c+\ell_s} = \mathbf{C}\hat{\mathbf{z}}_{t_c+\ell_s}$, $\hat{\mathbf{z}}_{t_c+\ell_s} = \mathbf{A}^{\ell_s}\boldsymbol{\mu}'_c$ ▷ Eq. (32)
return $\{\Theta', C', \hat{\mathbf{x}}_{t_c+\ell_s}\}$

Some Markov chains admit the existence of an invariant probability measure π that satisfies

$$\pi(B) = \int_{\mathcal{M}} \pi(dx)p(x,B), \quad B \in \Sigma_{\mathcal{M}}.$$

In this case, it is natural to take $\mathcal{F} = L^2_\pi(\mathcal{M})$ and denote the corresponding Koopman operator by $\mathcal{K}_\pi$. With the Hilbert space inner product

$$\langle \eta, \xi\rangle_{L^2_\pi(\mathcal{M})} \coloneqq \int_{\mathcal{M}} \eta(x)\overline{\xi(x)}\pi(dx),$$

the operator $\mathcal{K}_\pi$ is bounded on $L^2_\pi(\mathcal{M})$ and, by Jensen's inequality together with the invariant measure π, is a contraction:

$$\|\mathcal{K}_\pi\eta\|_{L^2_\pi(\mathcal{M})} \le \|\eta\|_{L^2_\pi(\mathcal{M})}.$$

Intuitively, the norm $\|\eta\|_{L^2_\pi(\mathcal{M})}$ measures the mean-square amplitude of the observable η when states are sampled according to the invariant measure π. The Koopman operator maps η to the one-step-ahead conditional mean $(\mathcal{K}_\pi\eta)(x)$. Thus, in stochastic systems, $\mathcal{K}_\pi$ acts as an averaging operator: it extracts the component of the future observable that is predictable from the current state. Such averaging can remove unpredictable fluctuations, but it cannot increase the mean-square amplitude under the stationary weighting π. Hence, $\mathcal{K}_\pi$ is non-expansive in $L^2_\pi(\mathcal{M})$. In addition,

Algorithm 2 StaticOptimization ($\mathbf{X}$)

Input: Time series data $\mathbf{X} \in \mathbb{R}^{d\times T}$
Output: Model Candidate $C = \{\boldsymbol{\theta}, \mathcal{S}, \mathcal{D}_T, \boldsymbol{\mu}_{T|T}, \mathbf{P}_{T|T}\}$

Initialize $\mathcal{D}_1 = \{\mathbf{x}_1\}$; $\mathcal{I}_1 = \{1\}$
for $t \in [T-1]$ **do**
  $\delta_{t+1} = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}_t^\top(\mathbf{x}_{t+1})\mathbf{K}_t^{-1}\mathbf{k}_t(\mathbf{x}_{t+1})$ ▷ Eq. (4)
  if $\delta_{t+1} > \nu$ **then**
    $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{\mathbf{x}_{t+1}\}$; $\mathcal{I}_{t+1} \leftarrow \mathcal{I}_t \cup \{t+1\}$
    Update $\mathbf{K}_t^{-1}$ to $\mathbf{K}_{t+1}^{-1}$ ▷ Eq. (5)
  end if
end for
Construct feature matrices Ψ, Ψ_0, Ψ_1 ▷ Eq. (6)
Compute moment matrices $\mathbf{S}_{00}, \mathbf{S}_{10}$ ▷ Eq. (8)
$\{\mathbf{U}_r, \Sigma_r, \mathbf{V}_r\} \leftarrow$ TruncateSVD($\mathbf{S}_{10}\mathbf{S}_{00}^{-1/2}$)
$\tilde{\mathbf{A}} \leftarrow \Sigma_r\mathbf{V}_r^\top\mathbf{S}_{00}^{-1/2}\mathbf{U}_r$, $\tilde{\mathbf{W}} \leftarrow \mathbf{U}_r$
$\tilde{\mathbf{C}} \leftarrow \mathbf{X}\mathbf{Z}^\top(\mathbf{Z}\mathbf{Z}^\top)^{-1}$ ▷ $\mathbf{Z} = \tilde{\mathbf{W}}^\dagger\Psi$
repeat
  Estimate $\{\boldsymbol{\mu}_{t|T}\}_{t=1}^T$, $\{\mathbf{P}_{t|T}\}_{t=1}^T$ ▷ Eqs. (9)-(16)
  Update $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{Q}, \mathbf{C}, \mathbf{W}, \mathbf{R}_x, \mathbf{R}_\psi, \boldsymbol{\mu}_0, \mathbf{P}_0\}$ ▷ Eqs. (17)-(22)
until convergence
Set $\mathcal{S} = \{\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3\}$ ▷ Definition 3
return $\{\boldsymbol{\theta}, \mathcal{S}, \mathcal{D}_T, \boldsymbol{\mu}_{T|T}, \mathbf{P}_{T|T}\}$

the classical Koopman operator for deterministic dynamical systems is recovered as a special case of this formulation. Specifically, for a measurable map $T : \mathcal{M} \to \mathcal{M}$ satisfying $X_{t+1} = T(X_t)$, setting $p(x, B) = \delta_{T(x)}(B)$ gives $(\mathcal{K}_\pi\eta)(x) = \eta(T(x))$, showing that the conditional-expectation formulation used here is a stochastic generalization of the classical deterministic Koopman framework.

One of the main advantages of Koopman operator theory is that it represents the evolution of observables through a linear operator, enabling spectral analysis even when the underlying state evolution is nonlinear. Indeed, in many situations, and notably for normal compact Koopman operators, the spectral theorem yields an orthonormal basis of a set of eigenfunctions $\{\psi_1, \psi_2, \ldots\}$ of $\mathcal{K}_\pi$ with eigenvalues $\{\lambda_1, \lambda_2, \ldots\}$ such that

$$\mathcal{K}_\pi\psi_i = \lambda_i\psi_i, \quad \forall i \in \mathbb{N},$$

where $\psi_i : \mathcal{M} \to \mathbb{C}$ and $\lambda_i \in \mathbb{C}$. Hence, every $\eta \in L^2_\pi(\mathcal{M})$ can be expressed as the superposition of simpler basis vectors as follows:

$$\eta = \sum_{i\in\mathbb{N}} \gamma_i\psi_i, \quad \gamma_i = \langle \eta, \psi_i\rangle_{L^2_\pi(\mathcal{M})}.$$

Consequently, for $t \in \mathbb{N}$,

$$(\mathcal{K}_\pi^t\eta) = \sum_{i\in\mathbb{N}} \lambda_i^t\gamma_i\psi_i,$$

which is known as Koopman mode decomposition (KMD) [8, 44]. For a fixed observable η, the decomposition is described by a set of triples $\{(\lambda_i, \psi_i, \gamma_i)\}_{i\in\mathbb{N}}$. However, a Koopman operator is not compact or normal in general. Therefore, without the above additional assumptions, there is no general guarantee that its eigenfunctions form a complete orthonormal basis of the space, which makes learning KMD challenging.

C Algorithm

Here, we collect pseudocode for our proposed method. Algorithm 1 outlines the overall streaming framework, which executes adaptive model selection, parameter updates, and real-time forecasting. Algorithm 2 details the static optimization procedure designed to estimate model parameters from a finite time series window.

D Proofs

D.1 Proof of Lemma 1

Lemma 1 (Time complexity of static optimization). *Let m be the final dictionary size produced by the basis orthogonalization, and let r be the latent dimension. Then, the time complexity of the static optimization is $O(Tm^2 + m^3 + \#iter \cdot Tr^2(d+m))$.*

Proof. We derive the time complexity of the static optimization described in Section 4.1. It consists of three stages; therefore, it suffices to sum the time complexities of each stage.

(1) Feature space basis orthogonalization. At each t, computing the residual δ_{t+1} in Eq. (3) requires forming the kernel vector $\mathbf{k}_t(\mathbf{x}_t) \in \mathbb{R}^{m_t}$ and a matrix–vector multiplication with $\mathbf{K}_t^{-1}$, which costs $O(m_t^2)$. If a point is added, updating $\mathbf{K}_{t+1}^{-1}$ by Eq. (5) is also $O(m_t^2)$. Since $m_t \leq m$ for all t, the total cost is $O(Tm^2)$.

(2) Regression-based initialization. Forming the moment matrices $\mathbf{S}_{00}$ and $\mathbf{S}_{10}$ in Eq. (8) costs $O(Tm^2)$. Computing $\mathbf{S}_{00}^{-1/2}$ via eigendecomposition costs $O(m^3)$. Subsequent multiplications to form $\mathbf{M} = \mathbf{S}_{10}\mathbf{S}_{00}^{-1/2}$ are $O(m^3)$ in the worst case, which is absorbed into $O(m^3)$. Finally, computing the truncated SVD of an $m \times m$ matrix to obtain the leading r components costs at most $O(m^2 r)$, which is also dominated by $O(m^3)$ because $r \leq m$. Hence, the second step costs $O(m^3 + Tm^2)$.

(3) Probabilistic refinement. Each EM iteration runs a Kalman filter forward pass and an RTS smoother backward pass for T steps. With latent dimension r and observation dimension $(d+m)$, if we compute the Kalman gain using the Woodbury identity, then the per-step cost is $O(r^2(d+m) + r^3) = O(r^2(d+m))$ because $r < m$. Therefore, the total cost per EM iteration is $O(Tr^2(d+m))$, and for $\#iter$ iterations it is $O(\#iter \cdot Tr^2(d+m))$. □

D.2 Proof of Lemma 2

Lemma 2 (Time complexity of AdaKoop). *Let m be the dictionary size, R be the number of stored models, and r be the latent dimension. The time complexity of AdaKoop is at least $O(m^2 + r^2(d+m))$ and at most $O(Tm^2 + m^3 + RTr^2(d+m) + \#iter \cdot Tr^2(d+m))$ per process.*

Proof. We analyze the time complexity per time step t_c in the streaming algorithm. AdaKoop first selects the best model for the current window $\mathbf{X}_c$ via statistical hypothesis testing. When the test does not trigger a switch, it continues to use the previous best model. It computes the residual δ_{t_c} and updates the dictionary inverse $\mathbf{K}^{-1}$ as shown in Eqs. (3) and (5), which takes $O(m^2)$. Subsequently, an online EM algorithm updates the sufficient statistics and parameters. By using the Woodbury matrix identity (similar to Lemma 1), the cost is dominated by matrix multiplications involving the latent dimension r and observation dimension $(d+m)$, resulting in $O(r^2(d+m))$. In contrast, when the test triggers a switch, AdaKoop determines to use other model. AdaKoop first takes $O(RTr^2(d+m))$ time to search for the better model in $\boldsymbol{\Theta}$. Furthermore, if AdaKoop encounters an unknown pattern, a new model is estimated from scratch using the static optimization on window $\mathbf{X}_c$. As proven in Lemma 1, the cost for this static optimization is $O(Tm^2 + m^3 + \#iter \cdot Tr^2(d+m))$. Finally, forecast an ℓ_s-steps-ahead future value via Eq. (32), which is negligible compared to the update steps. Summing these components yields the desired results. □

E Experiments

E.1 Experimental Setup

Computing infrastructure. We conducted all our experiments on a system running Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-105-generic x86_64), equipped with 2 ∗ Intel Xeon Gold 6444Y 3.60GHz 16-core CPU, 12 ∗ 64GB DDR5 RAM, and 2 ∗ 48GB NVIDIA RTX A6000 GPU.

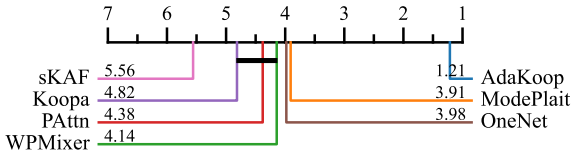
Benchmark. We adopted the dysts benchmark [23] as mentioned in the main text. The length of each dataset was 1,000 steps, with a time step of 0.01 and 5% noise ratio. We normalized the values of all datasets so that the maximum value was 1 and the minimum value was −1.

Implementation details. We now describe the implementation details we used throughout our experiments. The train/validation/test splits were 20%/10%/70%, and we set the length of the current window T at 100 steps. We conducted all our experiments with 5 different seeds for a fair comparison. We employed a grid search to determine each optimal hyperparameter such that it minimizes the error over the validation data. For AdaKoop, we used a standard Gaussian kernel where the kernel width was set to the median distance between the observed data. We set the threshold $\nu = 0.001$, the confidence level $\alpha = 0.01$, the switching limit $h = 3\chi_d^2(1-\alpha)$, and the number of iterations $\#iter = 3$. We varied the forgetting factor $\gamma \in \{0.01, 0.003, 0.001\}$ and the regularization parameter $\lambda_A \in \{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}$. For ModePlait, we varied the embedding dimension $h \in \{10, 20, 30\}$ and the threshold within $\{1.0, 1.5, 2.0\}$. For deep learning-based methods, we used the Adam optimizer [32] and varied the learning rate over $\{0.001, 0.003, 0.01, 0.03\}$.

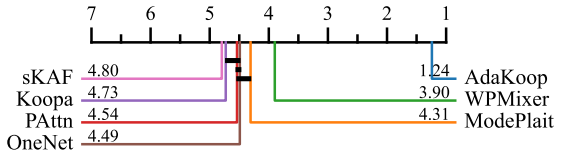
E.2 Additional Results

E.2.1 Q1. Accuracy. Here, we provide the detailed results for evaluating AdaKoop in terms of real-time forecasting. Figure 5 shows the critical difference diagrams for each metric. This diagram is based on the Wilcoxon-Holm method [57], where methods not connected by a bold line are sufficiently different regarding their average rank. It is clear that the significant improvements that AdaKoop achieved over its baselines are valid according to statistical tests. Moreover, Tables 5-13 show the mean and standard deviation of the results for all 71 datasets, where the best and second-best levels of performance are shown in **bold** and underlined, respectively. We can observe that AdaKoop achieved a comprehensive accuracy improvement over all its baselines.

E.2.2 Q3. Nonlinear capability. We considered four types of functions: the RBF kernel, the polynomial kernel, the sigmoid kernel,

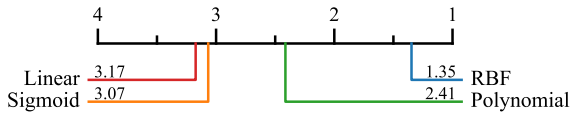


(a) Mean Squared Error (MSE)

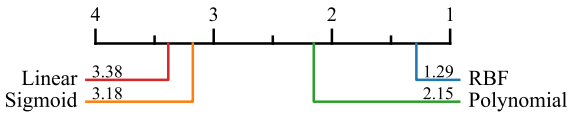


(b) Mean Absolute Error (MAE)

Figure 5: Critical difference diagrams of multivariate forecasting results. We used forecasting steps $\ell_s \in \{20, \ldots, 30\}$.



(a) Mean Squared Error (MSE)



(b) Mean Absolute Error (MAE)

Figure 6: Critical difference diagrams of AdaKoop with various kernels. We used forecasting steps $\ell_s \in \{20, \ldots, 30\}$.

and the linear kernel. These formulations are given by

$$k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{|\mathbf{x}-\mathbf{y}|^2}{2\sigma^2}\right),$$
$$k_{\mathrm{sigmoid}}(\mathbf{x}, \mathbf{y}) = \tanh\left(\gamma \mathbf{x}^\top \mathbf{y} + c_s\right),$$
$$k_{\mathrm{poly}}(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}^\top \mathbf{y} + c_p)^d,$$
$$k_{\mathrm{linear}}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y},$$

where $\sigma > 0$ is the length scale parameter, $\gamma > 0$ is the scale parameter, $c_p, c_s \geq 0$ are constants, and $d \in \mathbb{N}$ is the degree. We set σ as the median of the pairwise distances of the data matrix. For the polynomial and sigmoid kernels, we set $\gamma = 1/D$, where D is the number of dimensions. We used $d = 3$ and $c_p = 1$ for the polynomial kernel, and $c_s = 0$ for the sigmoid kernel. Figure 6 shows critical difference diagrams of AdaKoop with various kernel functions for each metric. Bold lines indicate insignificant differences between connected methods regarding their average rank. These results statistically demonstrate that AdaKoop effectively leverages the RKHS properties to estimate nonlinear dynamics from a nonstationary data stream.

F Limitations

While AdaKoop achieves remarkable improvements over its baselines, it has some limitations that may be worth addressing in the future. First, processing tensor streams with AdaKoop requires tensor data to be converted into a matricized or vectorized representation. Such a representation does not explicitly preserve the multi-way tensor structure in the model, which may limit its ability to capture multi-aspect relationships. Second, AdaKoop does not explicitly incorporate exogenous inputs or interventions. Consequently, changes caused by such external factors may be absorbed as regime shifts or process noise rather than being separated from intrinsic mechanisms. Extending AdaKoop toward controlled and intervention-aware dynamical modeling [35, 39] can broaden its applicability to clinical decision-making, closed-loop robotic control, and demand response forecasting. Lastly, our probabilistic model and change detection mechanism depend on linear Gaussian state space assumptions. As a result, data streams with extreme outliers or heavy-tailed noise may lead to degraded performance. Incorporating robust filtering methods based on generalized Bayes [14] or Student's t distribution [51] into AdaKoop is a promising direction.

Table 5: Detailed multivariate forecasting results with forecasting step $\ell_s = 20$

	Aizawa		Arneodo		Bouali2		BurkeShaw	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.019 ± 0.000**	**0.092 ± 0.000**	**0.028 ± 0.000**	**0.105 ± 0.000**	**0.057 ± 0.000**	**0.158 ± 0.000**	**0.095 ± 0.000**	**0.202 ± 0.000**
ModePlait	0.309 ± 0.015	0.445 ± 0.009	0.310 ± 0.042	0.417 ± 0.032	0.148 ± 0.002	0.288 ± 0.003	0.212 ± 0.003	0.356 ± 0.002
WPMixer	6.932 ± 13.67	0.876 ± 0.641	0.252 ± 0.230	0.305 ± 0.078	0.120 ± 0.006	0.237 ± 0.003	0.324 ± 0.328	0.398 ± 0.140
PAttn	0.604 ± 0.454	0.585 ± 0.188	0.142 ± 0.014	0.265 ± 0.012	0.168 ± 0.054	0.279 ± 0.031	0.300 ± 0.127	0.438 ± 0.085
Koopa	0.787 ± 0.155	0.629 ± 0.036	0.157 ± 0.012	0.295 ± 0.017	0.141 ± 0.017	0.255 ± 0.013	0.452 ± 0.254	0.489 ± 0.138
OneNet	0.426 ± 0.009	0.507 ± 0.004	0.487 ± 0.027	0.570 ± 0.013	0.125 ± 0.010	0.270 ± 0.016	0.236 ± 0.004	0.387 ± 0.005
sKAF	0.207 ± 0.028	0.236 ± 0.008	0.863 ± 0.305	0.359 ± 0.041	102.8 ± 60.78	3.060 ± 0.708	10.30 ± 2.122	0.815 ± 0.079

	Chen		ChenLee		Dadras		DequanLi	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.108 ± 0.000**	**0.245 ± 0.000**	**0.093 ± 0.000**	**0.191 ± 0.000**	**0.070 ± 0.000**	**0.185 ± 0.000**	**0.177 ± 0.000**	**0.296 ± 0.000**
ModePlait	0.338 ± 0.003	0.453 ± 0.005	0.266 ± 0.025	0.376 ± 0.015	0.146 ± 0.020	0.299 ± 0.022	0.296 ± 0.024	0.420 ± 0.021
WPMixer	0.423 ± 0.215	0.507 ± 0.121	0.213 ± 0.046	0.316 ± 0.028	0.105 ± 0.010	0.248 ± 0.011	1.763 ± 1.788	0.660 ± 0.224
PAttn	0.398 ± 0.075	0.516 ± 0.049	0.275 ± 0.157	0.368 ± 0.107	0.133 ± 0.013	0.280 ± 0.010	0.406 ± 0.284	0.370 ± 0.083
Koopa	0.627 ± 0.078	0.617 ± 0.034	0.197 ± 0.013	0.307 ± 0.016	0.123 ± 0.026	0.266 ± 0.028	0.465 ± 0.055	0.511 ± 0.026
OneNet	0.339 ± 0.065	0.478 ± 0.047	0.234 ± 0.024	0.318 ± 0.032	0.142 ± 0.022	0.304 ± 0.029	0.337 ± 0.050	0.475 ± 0.036
sKAF	0.217 ± 0.046	0.253 ± 0.015	25.80 ± 6.847	1.289 ± 0.164	38.01 ± 33.40	1.273 ± 0.526	306.1 ± 120.8	10.45 ± 1.841

	Finance		GenesioTesi		GuckenheimerHolmes		Hadley	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.211 ± 0.000**	**0.350 ± 0.000**	**0.006 ± 0.000**	**0.064 ± 0.000**	**0.038 ± 0.000**	**0.115 ± 0.000**	**0.017 ± 0.000**	**0.098 ± 0.000**
ModePlait	0.263 ± 0.010	0.420 ± 0.005	0.300 ± 0.006	0.413 ± 0.005	0.105 ± 0.018	0.214 ± 0.016	0.253 ± 0.034	0.382 ± 0.021
WPMixer	0.234 ± 0.033	0.367 ± 0.019	0.204 ± 0.006	0.364 ± 0.005	0.061 ± 0.016	0.155 ± 0.015	0.341 ± 0.232	0.390 ± 0.075
PAttn	0.287 ± 0.017	0.389 ± 0.008	0.231 ± 0.064	0.377 ± 0.066	0.117 ± 0.049	0.203 ± 0.034	0.229 ± 0.036	0.386 ± 0.027
Koopa	0.276 ± 0.043	0.400 ± 0.027	0.389 ± 0.293	0.438 ± 0.096	0.226 ± 0.061	0.258 ± 0.025	0.512 ± 0.187	0.514 ± 0.071
OneNet	0.313 ± 0.004	0.457 ± 0.004	0.438 ± 0.076	0.542 ± 0.049	0.190 ± 0.014	0.277 ± 0.012	0.247 ± 0.030	0.385 ± 0.015
sKAF	0.573 ± 0.060	0.422 ± 0.014	0.052 ± 0.035	0.154 ± 0.061	171.0 ± 9.877	6.144 ± 0.300	1.124 ± 0.569	0.532 ± 0.122

	Halvorsen		HenonHeiles		HyperBao		HyperCai	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.071 ± 0.000**	**0.204 ± 0.000**	**0.015 ± 0.000**	**0.085 ± 0.000**	**0.088 ± 0.000**	**0.217 ± 0.000**	**0.141 ± 0.000**	**0.277 ± 0.000**
ModePlait	0.333 ± 0.030	0.472 ± 0.022	0.368 ± 0.052	0.485 ± 0.040	0.215 ± 0.019	0.370 ± 0.019	0.317 ± 0.004	0.450 ± 0.004
WPMixer	0.509 ± 0.579	0.455 ± 0.246	0.123 ± 0.013	0.288 ± 0.016	0.315 ± 0.163	0.413 ± 0.103	0.562 ± 0.284	0.542 ± 0.145
PAttn	0.152 ± 0.015	0.316 ± 0.015	0.120 ± 0.047	0.278 ± 0.050	0.234 ± 0.041	0.381 ± 0.034	0.372 ± 0.135	0.479 ± 0.098
Koopa	0.217 ± 0.125	0.357 ± 0.085	0.177 ± 0.058	0.336 ± 0.050	0.236 ± 0.020	0.381 ± 0.012	0.440 ± 0.280	0.489 ± 0.141
OneNet	0.374 ± 0.029	0.505 ± 0.023	0.599 ± 0.082	0.666 ± 0.046	0.179 ± 0.005	0.347 ± 0.005	0.315 ± 0.039	0.464 ± 0.024
sKAF	0.267 ± 0.113	0.249 ± 0.031	3.439 ± 0.690	1.024 ± 0.103	11.70 ± 5.563	1.196 ± 0.214	34.00 ± 5.247	3.355 ± 0.414

	HyperJha		HyperLorenz		HyperLu		HyperPang	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.041 ± 0.000**	**0.144 ± 0.000**	**0.143 ± 0.000**	**0.268 ± 0.000**	**0.062 ± 0.000**	**0.177 ± 0.000**	**0.110 ± 0.000**	**0.241 ± 0.000**
ModePlait	0.297 ± 0.030	0.461 ± 0.021	0.242 ± 0.015	0.402 ± 0.016	0.213 ± 0.007	0.377 ± 0.006	0.222 ± 0.029	0.379 ± 0.027
WPMixer	0.925 ± 1.051	0.640 ± 0.260	0.260 ± 0.068	0.382 ± 0.034	0.268 ± 0.250	0.377 ± 0.124	0.625 ± 0.681	0.517 ± 0.214
PAttn	0.335 ± 0.157	0.466 ± 0.100	0.267 ± 0.021	0.384 ± 0.015	0.296 ± 0.183	0.416 ± 0.079	0.211 ± 0.033	0.374 ± 0.031
Koopa	0.318 ± 0.032	0.471 ± 0.028	0.301 ± 0.023	0.427 ± 0.014	0.383 ± 0.326	0.438 ± 0.150	0.223 ± 0.029	0.390 ± 0.026
OneNet	0.301 ± 0.050	0.451 ± 0.036	0.294 ± 0.023	0.458 ± 0.016	0.177 ± 0.021	0.347 ± 0.021	0.200 ± 0.034	0.361 ± 0.029
sKAF	0.417 ± 0.065	0.402 ± 0.028	4.279 ± 1.640	0.846 ± 0.080	2.524 ± 0.698	0.696 ± 0.076	11.98 ± 6.093	1.276 ± 0.341

	HyperQi		HyperRossler		HyperWang		HyperXu	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.060 ± 0.000**	**0.152 ± 0.000**	**0.052 ± 0.000**	**0.145 ± 0.000**	**0.169 ± 0.000**	**0.314 ± 0.000**	**0.185 ± 0.000**	**0.330 ± 0.000**
ModePlait	0.165 ± 0.016	0.291 ± 0.014	0.081 ± 0.013	0.183 ± 0.014	0.239 ± 0.031	0.370 ± 0.023	0.227 ± 0.027	0.376 ± 0.019
WPMixer	0.328 ± 0.276	0.314 ± 0.112	0.134 ± 0.048	0.194 ± 0.034	0.220 ± 0.007	0.369 ± 0.006	0.257 ± 0.114	0.387 ± 0.058
PAttn	0.272 ± 0.141	0.322 ± 0.037	0.075 ± 0.004	0.153 ± 0.008	0.349 ± 0.117	0.444 ± 0.072	0.278 ± 0.013	0.427 ± 0.011
Koopa	0.170 ± 0.065	0.285 ± 0.039	0.082 ± 0.011	0.168 ± 0.013	0.603 ± 0.369	0.495 ± 0.119	0.261 ± 0.012	0.408 ± 0.012
OneNet	0.154 ± 0.009	0.272 ± 0.012	0.106 ± 0.015	0.214 ± 0.033	0.186 ± 0.018	0.338 ± 0.012	0.212 ± 0.019	0.376 ± 0.019
sKAF	4.048 ± 4.547	0.755 ± 0.297	239.4 ± 105.5	7.499 ± 1.700	32.16 ± 9.755	2.904 ± 0.450	23.41 ± 10.01	2.369 ± 0.567

Table 6: Detailed multivariate forecasting results with forecasting step $\ell_s = 20$

	HyperYan		HyperYangChen		KawczynskiStrizhak		Laser	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.069 ± 0.000**	**0.198 ± 0.000**	**0.117 ± 0.000**	**0.252 ± 0.000**	**0.034 ± 0.000**	**0.111 ± 0.000**	**0.083 ± 0.000**	**0.197 ± 0.000**
ModePlait	0.227 ± 0.006	0.367 ± 0.008	0.288 ± 0.007	0.445 ± 0.005	0.037 ± 0.000	0.117 ± 0.002	0.324 ± 0.009	0.447 ± 0.007
WPMixer	0.311 ± 0.172	0.408 ± 0.084	0.227 ± 0.039	0.386 ± 0.032	0.039 ± 0.002	0.111 ± 0.003	0.503 ± 0.357	0.496 ± 0.148
PAttn	0.333 ± 0.137	0.449 ± 0.105	0.363 ± 0.080	0.489 ± 0.041	0.044 ± 0.002	0.119 ± 0.004	0.401 ± 0.144	0.492 ± 0.064
Koopa	0.349 ± 0.081	0.448 ± 0.059	0.352 ± 0.044	0.477 ± 0.033	0.045 ± 0.001	0.119 ± 0.002	0.393 ± 0.295	0.474 ± 0.138
OneNet	0.224 ± 0.056	0.369 ± 0.052	0.258 ± 0.016	0.426 ± 0.014	0.046 ± 0.001	0.133 ± 0.002	0.360 ± 0.005	0.477 ± 0.004
sKAF	3.222 ± 3.001	0.588 ± 0.133	3.289 ± 2.577	0.808 ± 0.150	0.062 ± 0.012	0.139 ± 0.012	39.54 ± 18.34	3.718 ± 0.958

	Lorenz		LorenzBounded		LorenzStenflo		LuChen	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.142 ± 0.000**	**0.295 ± 0.000**	**0.088 ± 0.000**	**0.206 ± 0.000**	**0.229 ± 0.000**	**0.360 ± 0.000**	0.099 ± 0.000	0.255 ± 0.000
ModePlait	0.267 ± 0.028	0.421 ± 0.024	0.360 ± 0.017	0.490 ± 0.010	0.280 ± 0.025	0.432 ± 0.018	0.217 ± 0.018	0.375 ± 0.020
WPMixer	0.556 ± 0.428	0.543 ± 0.192	0.530 ± 0.554	0.485 ± 0.147	0.307 ± 0.143	0.434 ± 0.078	0.045 ± 0.007	0.164 ± 0.009
PAttn	0.260 ± 0.022	0.405 ± 0.016	0.784 ± 0.285	0.626 ± 0.123	0.306 ± 0.099	0.437 ± 0.055	**0.040 ± 0.004**	**0.153 ± 0.006**
Koopa	0.402 ± 0.218	0.467 ± 0.086	0.432 ± 0.016	0.500 ± 0.013	0.360 ± 0.153	0.470 ± 0.083	0.063 ± 0.005	0.188 ± 0.007
OneNet	0.342 ± 0.053	0.471 ± 0.050	0.320 ± 0.060	0.480 ± 0.046	0.272 ± 0.028	0.423 ± 0.025	0.294 ± 0.022	0.451 ± 0.018
sKAF	0.244 ± 0.072	0.298 ± 0.033	3.141 ± 3.654	0.440 ± 0.187	0.378 ± 0.064	0.382 ± 0.022	0.112 ± 0.023	0.225 ± 0.012

	LuChenCheng		MooreSpiegel		NewtonLiepnik		NoseHoover	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	0.013 ± 0.000	0.083 ± 0.000	**0.078 ± 0.000**	**0.216 ± 0.000**	**0.143 ± 0.000**	**0.264 ± 0.000**	**0.111 ± 0.000**	**0.221 ± 0.000**
ModePlait	0.408 ± 0.028	0.496 ± 0.022	0.230 ± 0.016	0.371 ± 0.019	0.241 ± 0.032	0.383 ± 0.030	0.349 ± 0.010	0.459 ± 0.008
WPMixer	0.706 ± 1.017	0.492 ± 0.218	0.142 ± 0.011	0.262 ± 0.005	0.197 ± 0.010	0.355 ± 0.010	0.184 ± 0.010	0.305 ± 0.009
PAttn	0.349 ± 0.028	0.479 ± 0.027	0.142 ± 0.013	0.269 ± 0.005	0.264 ± 0.076	0.403 ± 0.053	0.207 ± 0.009	0.321 ± 0.008
Koopa	0.465 ± 0.061	0.522 ± 0.031	0.160 ± 0.006	0.287 ± 0.008	0.197 ± 0.003	0.351 ± 0.005	0.214 ± 0.039	0.342 ± 0.032
OneNet	0.434 ± 0.007	0.533 ± 0.006	0.247 ± 0.031	0.395 ± 0.026	0.235 ± 0.037	0.395 ± 0.037	0.339 ± 0.109	0.472 ± 0.075
sKAF	**0.008 ± 0.001**	**0.064 ± 0.003**	0.149 ± 0.029	0.242 ± 0.007	1.429 ± 0.127	0.439 ± 0.010	0.398 ± 0.360	0.341 ± 0.107

	NuclearQuadrupole		PehlivanWei		Qi		QiChen	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.032 ± 0.000**	**0.128 ± 0.000**	**0.058 ± 0.000**	**0.156 ± 0.000**	**0.144 ± 0.000**	**0.263 ± 0.000**	**0.068 ± 0.000**	**0.212 ± 0.000**
ModePlait	0.427 ± 0.026	0.532 ± 0.017	0.247 ± 0.021	0.369 ± 0.028	0.276 ± 0.035	0.424 ± 0.023	0.289 ± 0.007	0.450 ± 0.007
WPMixer	0.657 ± 0.687	0.560 ± 0.166	0.168 ± 0.019	0.302 ± 0.022	0.489 ± 0.209	0.538 ± 0.088	0.512 ± 0.562	0.522 ± 0.238
PAttn	0.451 ± 0.235	0.529 ± 0.111	0.269 ± 0.049	0.406 ± 0.045	0.552 ± 0.032	0.615 ± 0.017	0.710 ± 0.243	0.662 ± 0.138
Koopa	0.323 ± 0.013	0.450 ± 0.009	0.486 ± 0.463	0.467 ± 0.231	0.728 ± 0.093	0.612 ± 0.039	0.335 ± 0.031	0.453 ± 0.017
OneNet	0.457 ± 0.047	0.576 ± 0.031	0.222 ± 0.059	0.364 ± 0.053	0.279 ± 0.029	0.415 ± 0.011	0.282 ± 0.062	0.443 ± 0.058
sKAF	0.995 ± 0.169	0.597 ± 0.038	14.79 ± 8.505	1.113 ± 0.207	1.115 ± 0.702	0.427 ± 0.078	0.476 ± 0.276	0.315 ± 0.044

	RabinovichFabrikant		RayleighBenard		RikitakeDynamo		Rossler	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.080 ± 0.000**	**0.223 ± 0.000**	**0.134 ± 0.000**	**0.255 ± 0.000**	**0.150 ± 0.000**	**0.279 ± 0.000**	**0.015 ± 0.000**	**0.075 ± 0.000**
ModePlait	0.278 ± 0.018	0.422 ± 0.017	0.275 ± 0.024	0.432 ± 0.016	0.357 ± 0.031	0.482 ± 0.028	0.300 ± 0.056	0.403 ± 0.035
WPMixer	0.215 ± 0.111	0.349 ± 0.109	0.591 ± 0.722	0.531 ± 0.243	0.364 ± 0.250	0.410 ± 0.149	0.144 ± 0.023	0.270 ± 0.020
PAttn	0.335 ± 0.116	0.460 ± 0.099	0.340 ± 0.173	0.478 ± 0.113	0.328 ± 0.067	0.418 ± 0.072	0.231 ± 0.106	0.340 ± 0.092
Koopa	0.488 ± 0.343	0.528 ± 0.141	0.319 ± 0.021	0.455 ± 0.012	0.233 ± 0.011	0.352 ± 0.034	0.404 ± 0.272	0.410 ± 0.119
OneNet	0.489 ± 0.044	0.604 ± 0.026	0.302 ± 0.047	0.436 ± 0.033	0.373 ± 0.055	0.491 ± 0.040	0.437 ± 0.137	0.517 ± 0.073
sKAF	0.181 ± 0.099	0.273 ± 0.062	0.971 ± 0.191	0.503 ± 0.027	122.8 ± 138.2	3.374 ± 1.957	0.023 ± 0.016	0.075 ± 0.010

	Rucklidge		Sakarya		ShimizuMorioka		SprottA	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.096 ± 0.000**	**0.242 ± 0.000**	**0.050 ± 0.000**	**0.161 ± 0.000**	**0.101 ± 0.000**	**0.226 ± 0.000**	**0.083 ± 0.000**	**0.202 ± 0.000**
ModePlait	0.333 ± 0.025	0.481 ± 0.018	0.294 ± 0.025	0.423 ± 0.021	0.411 ± 0.023	0.540 ± 0.014	0.389 ± 0.025	0.500 ± 0.018
WPMixer	0.236 ± 0.030	0.388 ± 0.024	0.359 ± 0.361	0.413 ± 0.151	0.310 ± 0.017	0.451 ± 0.014	0.092 ± 0.016	0.240 ± 0.024
PAttn	0.998 ± 0.126	0.769 ± 0.057	0.270 ± 0.078	0.410 ± 0.062	0.442 ± 0.086	0.546 ± 0.066	0.098 ± 0.008	0.256 ± 0.013
Koopa	1.251 ± 0.535	0.792 ± 0.154	0.465 ± 0.314	0.479 ± 0.140	0.824 ± 0.413	0.678 ± 0.144	0.130 ± 0.022	0.279 ± 0.027
OneNet	0.388 ± 0.053	0.514 ± 0.040	0.344 ± 0.072	0.483 ± 0.049	0.382 ± 0.007	0.515 ± 0.006	0.355 ± 0.064	0.493 ± 0.047
sKAF	0.595 ± 0.170	0.354 ± 0.032	0.092 ± 0.011	0.204 ± 0.007	0.157 ± 0.069	0.244 ± 0.025	1.400 ± 0.544	0.620 ± 0.102

Table 7: Detailed multivariate forecasting results with forecasting step $\ell_s = 20$

	SprottB		SprottC		SprottD		SprottE	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.075 ± 0.000**	**0.191 ± 0.000**	**0.077 ± 0.000**	**0.176 ± 0.000**	**0.041 ± 0.000**	**0.150 ± 0.000**	**0.122 ± 0.000**	**0.240 ± 0.000**
ModePlait	0.288 ± 0.025	0.433 ± 0.022	0.233 ± 0.020	0.352 ± 0.013	0.187 ± 0.006	0.330 ± 0.008	0.291 ± 0.005	0.417 ± 0.008
WPMixer	1.089 ± 0.486	0.689 ± 0.133	0.219 ± 0.076	0.344 ± 0.053	0.132 ± 0.018	0.274 ± 0.019	1.023 ± 1.118	0.560 ± 0.239
PAttn	0.365 ± 0.036	0.492 ± 0.024	0.298 ± 0.111	0.401 ± 0.073	0.211 ± 0.059	0.339 ± 0.042	0.648 ± 0.242	0.605 ± 0.129
Koopa	0.617 ± 0.320	0.577 ± 0.113	0.444 ± 0.181	0.447 ± 0.093	0.281 ± 0.084	0.377 ± 0.055	0.597 ± 0.317	0.537 ± 0.134
OneNet	0.274 ± 0.017	0.409 ± 0.012	0.177 ± 0.003	0.302 ± 0.005	0.159 ± 0.002	0.316 ± 0.002	0.238 ± 0.038	0.386 ± 0.027
sKAF	1.393 ± 0.815	0.526 ± 0.133	478.4 ± 566.3	5.309 ± 3.437	131.1 ± 19.32	4.271 ± 0.344	0.722 ± 0.493	0.322 ± 0.061

	SprottF		SprottG		SprottH		SprottI	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.038 ± 0.000**	**0.146 ± 0.000**	0.008 ± 0.000	0.063 ± 0.000	**0.040 ± 0.000**	**0.141 ± 0.000**	0.017 ± 0.000	0.096 ± 0.000
ModePlait	0.248 ± 0.015	0.390 ± 0.013	0.306 ± 0.060	0.434 ± 0.042	0.280 ± 0.013	0.417 ± 0.012	0.447 ± 0.036	0.526 ± 0.041
WPMixer	0.260 ± 0.077	0.407 ± 0.050	1.216 ± 2.201	0.547 ± 0.411	0.215 ± 0.037	0.367 ± 0.032	0.156 ± 0.033	0.309 ± 0.028
PAttn	0.272 ± 0.049	0.422 ± 0.045	0.621 ± 0.042	0.560 ± 0.011	0.452 ± 0.163	0.530 ± 0.110	0.240 ± 0.083	0.394 ± 0.074
Koopa	0.530 ± 0.217	0.508 ± 0.035	0.185 ± 0.023	0.309 ± 0.018	0.263 ± 0.101	0.398 ± 0.053	0.562 ± 0.581	0.467 ± 0.243
OneNet	0.261 ± 0.069	0.416 ± 0.051	0.427 ± 0.105	0.498 ± 0.071	0.170 ± 0.002	0.333 ± 0.004	0.438 ± 0.072	0.549 ± 0.042
sKAF	0.629 ± 0.269	0.265 ± 0.053	**0.006 ± 0.002**	**0.046 ± 0.003**	0.394 ± 0.184	0.255 ± 0.040	**0.015 ± 0.002**	**0.079 ± 0.004**

	SprottJ		SprottJerk		SprottK		SprottL	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.009 ± 0.000**	**0.068 ± 0.000**	**0.027 ± 0.000**	0.110 ± 0.000	0.014 ± 0.000	0.077 ± 0.000	0.062 ± 0.000	0.187 ± 0.000
ModePlait	0.405 ± 0.062	0.507 ± 0.040	0.423 ± 0.055	0.512 ± 0.035	0.399 ± 0.043	0.494 ± 0.030	0.291 ± 0.042	0.434 ± 0.031
WPMixer	0.165 ± 0.012	0.327 ± 0.016	0.268 ± 0.282	0.322 ± 0.178	0.231 ± 0.020	0.377 ± 0.019	0.126 ± 0.006	0.268 ± 0.013
PAttn	0.442 ± 0.125	0.535 ± 0.084	0.079 ± 0.033	0.214 ± 0.054	0.251 ± 0.012	0.395 ± 0.016	0.137 ± 0.037	0.284 ± 0.032
Koopa	1.185 ± 1.847	0.664 ± 0.422	0.073 ± 0.024	0.209 ± 0.041	0.273 ± 0.107	0.382 ± 0.060	0.362 ± 0.211	0.417 ± 0.107
OneNet	0.499 ± 0.060	0.584 ± 0.023	0.621 ± 0.038	0.660 ± 0.022	0.481 ± 0.008	0.573 ± 0.008	0.487 ± 0.021	0.588 ± 0.011
sKAF	0.055 ± 0.004	0.124 ± 0.005	0.031 ± 0.005	**0.093 ± 0.003**	**0.008 ± 0.000**	**0.066 ± 0.001**	**0.050 ± 0.019**	**0.153 ± 0.017**

	SprottM		SprottN		SprottO		SprottP	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.008 ± 0.000**	**0.065 ± 0.000**	**0.008 ± 0.000**	**0.071 ± 0.000**	0.012 ± 0.000	0.087 ± 0.000	**0.083 ± 0.000**	**0.190 ± 0.000**
ModePlait	0.345 ± 0.002	0.473 ± 0.004	0.346 ± 0.018	0.468 ± 0.014	0.456 ± 0.024	0.513 ± 0.032	0.245 ± 0.027	0.382 ± 0.023
WPMixer	5.098 ± 3.743	1.374 ± 0.622	0.374 ± 0.250	0.452 ± 0.126	2.139 ± 4.717	0.361 ± 0.500	0.366 ± 0.116	0.444 ± 0.049
PAttn	0.565 ± 0.260	0.591 ± 0.136	0.269 ± 0.014	0.410 ± 0.009	0.028 ± 0.006	0.130 ± 0.018	0.416 ± 0.026	0.485 ± 0.009
Koopa	1.322 ± 1.354	0.760 ± 0.262	0.422 ± 0.086	0.516 ± 0.048	0.053 ± 0.015	0.186 ± 0.028	0.401 ± 0.043	0.458 ± 0.020
OneNet	0.532 ± 0.002	0.594 ± 0.003	0.520 ± 0.002	0.593 ± 0.004	0.867 ± 0.048	0.817 ± 0.022	0.341 ± 0.006	0.478 ± 0.005
sKAF	0.034 ± 0.014	0.100 ± 0.010	0.012 ± 0.004	0.074 ± 0.009	**0.005 ± 0.000**	**0.054 ± 0.001**	1.159 ± 0.379	0.439 ± 0.087

	SprottQ		SprottR		SprottS		SprottTorus	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.039 ± 0.000**	**0.118 ± 0.000**	**0.033 ± 0.000**	**0.101 ± 0.000**	**0.019 ± 0.000**	**0.105 ± 0.000**	**0.085 ± 0.000**	**0.161 ± 0.000**
ModePlait	0.400 ± 0.025	0.491 ± 0.010	0.201 ± 0.011	0.331 ± 0.007	0.405 ± 0.054	0.499 ± 0.045	0.219 ± 0.005	0.336 ± 0.006
WPMixer	0.324 ± 0.237	0.439 ± 0.121	0.353 ± 0.127	0.398 ± 0.065	0.605 ± 0.398	0.523 ± 0.164	0.347 ± 0.205	0.402 ± 0.109
PAttn	0.325 ± 0.037	0.431 ± 0.027	0.367 ± 0.031	0.427 ± 0.017	0.339 ± 0.142	0.427 ± 0.079	0.419 ± 0.147	0.459 ± 0.093
Koopa	0.893 ± 0.907	0.606 ± 0.264	0.811 ± 0.376	0.555 ± 0.121	0.757 ± 0.711	0.579 ± 0.212	0.352 ± 0.133	0.418 ± 0.038
OneNet	0.484 ± 0.004	0.539 ± 0.005	0.266 ± 0.034	0.400 ± 0.023	0.436 ± 0.069	0.541 ± 0.045	0.188 ± 0.003	0.294 ± 0.002
sKAF	1.360 ± 0.382	0.483 ± 0.075	1.926 ± 3.368	0.222 ± 0.151	4.035 ± 1.429	0.970 ± 0.122	12.15 ± 2.799	0.530 ± 0.045

	VallisElNino		WangSun		ZhouChen	
	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	0.195 ± 0.000	0.327 ± 0.000	0.087 ± 0.000	0.205 ± 0.000	**0.047 ± 0.000**	**0.148 ± 0.000**
ModePlait	0.190 ± 0.016	0.350 ± 0.010	0.098 ± 0.010	0.222 ± 0.013	0.127 ± 0.022	0.226 ± 0.019
WPMixer	0.206 ± 0.036	0.363 ± 0.019	0.153 ± 0.125	0.244 ± 0.056	1.380 ± 2.774	0.275 ± 0.128
PAttn	0.240 ± 0.011	0.392 ± 0.009	0.129 ± 0.009	0.264 ± 0.011	0.208 ± 0.079	0.288 ± 0.056
Koopa	0.234 ± 0.010	0.385 ± 0.006	0.166 ± 0.126	0.261 ± 0.064	0.236 ± 0.103	0.262 ± 0.027
OneNet	0.230 ± 0.005	0.387 ± 0.004	**0.073 ± 0.004**	**0.191 ± 0.011**	0.143 ± 0.070	0.222 ± 0.031
sKAF	**0.184 ± 0.015**	**0.318 ± 0.005**	427.0 ± 94.07	4.331 ± 0.347	130.3 ± 97.99	2.775 ± 1.133

Table 8: Detailed multivariate forecasting results with forecasting step $\ell_s = 25$

	Aizawa		Arneodo		Bouali2		BurkeShaw	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.027 ± 0.000**	**0.112 ± 0.000**	**0.035 ± 0.000**	**0.122 ± 0.000**	**0.073 ± 0.000**	**0.181 ± 0.000**	**0.142 ± 0.000**	**0.268 ± 0.000**
ModePlait	0.342 ± 0.018	0.462 ± 0.010	0.336 ± 0.031	0.437 ± 0.022	0.153 ± 0.003	0.295 ± 0.002	0.227 ± 0.002	0.368 ± 0.001
WPMixer	5.689 ± 10.74	0.881 ± 0.599	0.266 ± 0.228	0.322 ± 0.076	0.125 ± 0.003	0.244 ± 0.006	0.271 ± 0.209	0.384 ± 0.097
PAttn	0.593 ± 0.341	0.590 ± 0.153	0.159 ± 0.009	0.279 ± 0.006	0.183 ± 0.047	0.284 ± 0.025	0.320 ± 0.116	0.458 ± 0.078
Koopa	0.824 ± 0.076	0.666 ± 0.024	0.159 ± 0.017	0.293 ± 0.013	0.150 ± 0.006	0.267 ± 0.004	0.470 ± 0.275	0.501 ± 0.152
OneNet	0.432 ± 0.014	0.510 ± 0.007	0.497 ± 0.025	0.577 ± 0.009	0.125 ± 0.008	0.267 ± 0.009	0.240 ± 0.001	0.391 ± 0.002
sKAF	0.312 ± 0.055	0.283 ± 0.016	0.631 ± 0.350	0.349 ± 0.075	130.3 ± 82.01	3.257 ± 0.814	20.77 ± 20.33	1.101 ± 0.357

	Chen		ChenLee		Dadras		DequanLi	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.172 ± 0.000**	**0.320 ± 0.000**	**0.156 ± 0.000**	**0.250 ± 0.000**	**0.087 ± 0.000**	**0.214 ± 0.000**	**0.255 ± 0.000**	**0.359 ± 0.000**
ModePlait	0.344 ± 0.010	0.459 ± 0.007	0.267 ± 0.025	0.383 ± 0.016	0.150 ± 0.021	0.305 ± 0.025	0.319 ± 0.036	0.435 ± 0.028
WPMixer	0.404 ± 0.170	0.507 ± 0.102	0.221 ± 0.030	0.333 ± 0.028	0.105 ± 0.008	0.246 ± 0.008	2.064 ± 2.232	0.701 ± 0.226
PAttn	0.397 ± 0.066	0.515 ± 0.037	0.300 ± 0.178	0.388 ± 0.126	0.137 ± 0.012	0.283 ± 0.012	0.475 ± 0.406	0.379 ± 0.088
Koopa	0.638 ± 0.048	0.604 ± 0.015	0.215 ± 0.018	0.331 ± 0.016	0.128 ± 0.033	0.270 ± 0.032	0.482 ± 0.086	0.528 ± 0.043
OneNet	0.342 ± 0.066	0.480 ± 0.047	0.249 ± 0.023	0.325 ± 0.038	0.141 ± 0.023	0.300 ± 0.030	0.336 ± 0.048	0.475 ± 0.032
sKAF	0.554 ± 0.344	0.371 ± 0.046	37.89 ± 7.978	1.616 ± 0.128	24.21 ± 13.00	1.264 ± 0.274	266.9 ± 220.0	9.483 ± 2.536

	Finance		GenesioTesi		GuckenheimerHolmes		Hadley	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.207 ± 0.000**	**0.350 ± 0.000**	**0.009 ± 0.000**	**0.075 ± 0.000**	**0.047 ± 0.000**	**0.131 ± 0.000**	**0.029 ± 0.000**	**0.129 ± 0.000**
ModePlait	0.271 ± 0.018	0.427 ± 0.007	0.313 ± 0.009	0.425 ± 0.007	0.121 ± 0.025	0.228 ± 0.022	0.275 ± 0.032	0.399 ± 0.018
WPMixer	0.220 ± 0.019	0.359 ± 0.009	0.255 ± 0.032	0.411 ± 0.025	0.085 ± 0.022	0.177 ± 0.009	0.471 ± 0.284	0.459 ± 0.079
PAttn	0.292 ± 0.019	0.399 ± 0.015	0.281 ± 0.070	0.422 ± 0.070	0.166 ± 0.063	0.240 ± 0.028	0.261 ± 0.052	0.411 ± 0.034
Koopa	0.252 ± 0.032	0.389 ± 0.021	0.539 ± 0.465	0.486 ± 0.113	0.289 ± 0.108	0.290 ± 0.031	0.551 ± 0.220	0.541 ± 0.090
OneNet	0.307 ± 0.002	0.450 ± 0.003	0.411 ± 0.099	0.527 ± 0.064	0.191 ± 0.010	0.278 ± 0.010	0.243 ± 0.040	0.383 ± 0.022
sKAF	0.783 ± 0.078	0.478 ± 0.022	0.056 ± 0.046	0.152 ± 0.054	131.7 ± 48.32	5.004 ± 0.977	1.781 ± 0.669	0.639 ± 0.113

	Halvorsen		HenonHeiles		HyperBao		HyperCai	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.100 ± 0.000**	**0.239 ± 0.000**	**0.021 ± 0.000**	**0.104 ± 0.000**	**0.111 ± 0.000**	**0.248 ± 0.000**	**0.214 ± 0.000**	**0.332 ± 0.000**
ModePlait	0.367 ± 0.036	0.492 ± 0.027	0.445 ± 0.052	0.538 ± 0.035	0.233 ± 0.021	0.384 ± 0.018	0.332 ± 0.004	0.460 ± 0.002
WPMixer	0.429 ± 0.467	0.429 ± 0.202	0.128 ± 0.014	0.296 ± 0.018	0.336 ± 0.167	0.433 ± 0.099	0.532 ± 0.227	0.542 ± 0.114
PAttn	0.157 ± 0.013	0.317 ± 0.013	0.123 ± 0.021	0.288 ± 0.025	0.259 ± 0.035	0.403 ± 0.033	0.403 ± 0.120	0.495 ± 0.084
Koopa	0.228 ± 0.080	0.374 ± 0.057	0.174 ± 0.054	0.331 ± 0.047	0.264 ± 0.040	0.403 ± 0.029	0.435 ± 0.275	0.487 ± 0.128
OneNet	0.373 ± 0.036	0.502 ± 0.027	0.614 ± 0.071	0.675 ± 0.040	0.184 ± 0.002	0.352 ± 0.003	0.321 ± 0.042	0.467 ± 0.028
sKAF	0.277 ± 0.057	0.302 ± 0.015	2.246 ± 0.558	0.796 ± 0.078	8.502 ± 3.907	1.152 ± 0.234	34.54 ± 25.67	3.207 ± 0.845

	HyperJha		HyperLorenz		HyperLu		HyperPang	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.057 ± 0.000**	**0.172 ± 0.000**	**0.147 ± 0.000**	**0.280 ± 0.000**	**0.075 ± 0.000**	**0.199 ± 0.000**	**0.145 ± 0.000**	**0.286 ± 0.000**
ModePlait	0.309 ± 0.043	0.471 ± 0.029	0.262 ± 0.016	0.417 ± 0.015	0.221 ± 0.007	0.384 ± 0.006	0.237 ± 0.031	0.387 ± 0.028
WPMixer	0.851 ± 0.875	0.633 ± 0.221	0.261 ± 0.057	0.386 ± 0.025	0.270 ± 0.233	0.379 ± 0.109	0.491 ± 0.431	0.491 ± 0.169
PAttn	0.338 ± 0.133	0.469 ± 0.089	0.302 ± 0.021	0.410 ± 0.011	0.318 ± 0.211	0.425 ± 0.095	0.228 ± 0.039	0.386 ± 0.035
Koopa	0.319 ± 0.018	0.472 ± 0.014	0.291 ± 0.009	0.422 ± 0.006	0.436 ± 0.397	0.459 ± 0.169	0.228 ± 0.036	0.394 ± 0.033
OneNet	0.289 ± 0.062	0.444 ± 0.049	0.292 ± 0.024	0.456 ± 0.017	0.172 ± 0.023	0.340 ± 0.027	0.205 ± 0.038	0.364 ± 0.032
sKAF	1.064 ± 0.362	0.582 ± 0.058	8.469 ± 1.321	1.221 ± 0.027	4.471 ± 1.562	0.922 ± 0.132	26.83 ± 12.45	1.962 ± 0.359

	HyperQi		HyperRossler		HyperWang		HyperXu	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.100 ± 0.000**	**0.197 ± 0.000**	**0.063 ± 0.000**	**0.164 ± 0.000**	0.227 ± 0.000	0.358 ± 0.000	**0.210 ± 0.000**	**0.358 ± 0.000**
ModePlait	0.173 ± 0.013	0.297 ± 0.014	0.088 ± 0.014	0.189 ± 0.015	0.245 ± 0.036	0.375 ± 0.029	0.242 ± 0.029	0.388 ± 0.021
WPMixer	0.404 ± 0.361	0.337 ± 0.112	0.144 ± 0.054	0.208 ± 0.039	0.221 ± 0.016	0.376 ± 0.010	0.256 ± 0.096	0.388 ± 0.049
PAttn	0.267 ± 0.128	0.327 ± 0.038	0.085 ± 0.007	0.164 ± 0.005	0.374 ± 0.128	0.472 ± 0.077	0.294 ± 0.021	0.435 ± 0.019
Koopa	0.205 ± 0.104	0.310 ± 0.054	0.090 ± 0.015	0.174 ± 0.014	0.595 ± 0.361	0.504 ± 0.112	0.267 ± 0.018	0.407 ± 0.014
OneNet	0.161 ± 0.022	0.281 ± 0.025	0.117 ± 0.021	0.226 ± 0.036	**0.180 ± 0.018**	**0.334 ± 0.013**	0.217 ± 0.017	0.376 ± 0.017
sKAF	4.296 ± 5.137	0.825 ± 0.346	541.1 ± 243.2	7.696 ± 1.884	98.90 ± 43.41	4.344 ± 0.717	30.47 ± 11.02	2.642 ± 0.599

Table 9: Detailed multivariate forecasting results with forecasting step $\ell_s = 25$

	HyperYan		HyperYangChen		KawczynskiStrizhak		Laser	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.118 ± 0.000**	**0.261 ± 0.000**	**0.178 ± 0.000**	**0.319 ± 0.000**	**0.046 ± 0.000**	0.121 ± 0.000	**0.125 ± 0.000**	**0.248 ± 0.000**
ModePlait	0.241 ± 0.006	0.382 ± 0.005	0.310 ± 0.006	0.461 ± 0.007	0.050 ± 0.000	0.129 ± 0.002	0.345 ± 0.009	0.468 ± 0.007
WPMixer	0.270 ± 0.088	0.395 ± 0.058	0.256 ± 0.038	0.411 ± 0.029	0.051 ± 0.003	**0.121 ± 0.003**	0.508 ± 0.376	0.511 ± 0.160
PAttn	0.322 ± 0.116	0.445 ± 0.095	0.377 ± 0.059	0.502 ± 0.027	0.058 ± 0.004	0.127 ± 0.008	0.409 ± 0.116	0.505 ± 0.065
Koopa	0.335 ± 0.074	0.443 ± 0.048	0.376 ± 0.041	0.491 ± 0.029	0.056 ± 0.002	0.125 ± 0.002	0.341 ± 0.203	0.448 ± 0.105
OneNet	0.233 ± 0.058	0.377 ± 0.053	0.263 ± 0.024	0.429 ± 0.021	0.058 ± 0.000	0.142 ± 0.001	0.372 ± 0.003	0.489 ± 0.003
sKAF	3.659 ± 1.170	0.713 ± 0.037	7.862 ± 6.727	1.284 ± 0.423	0.070 ± 0.018	0.148 ± 0.017	82.58 ± 80.64	4.788 ± 2.302

	Lorenz		LorenzBounded		LorenzStenflo		LuChen	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.180 ± 0.000**	**0.331 ± 0.000**	**0.115 ± 0.000**	**0.244 ± 0.000**	**0.262 ± 0.000**	**0.401 ± 0.000**	0.120 ± 0.000	0.286 ± 0.000
ModePlait	0.293 ± 0.025	0.440 ± 0.022	0.385 ± 0.015	0.512 ± 0.010	0.296 ± 0.023	0.447 ± 0.018	0.228 ± 0.016	0.381 ± 0.018
WPMixer	0.466 ± 0.297	0.502 ± 0.136	0.437 ± 0.313	0.475 ± 0.102	0.302 ± 0.101	0.430 ± 0.058	0.049 ± 0.009	0.167 ± 0.018
PAttn	0.273 ± 0.023	0.412 ± 0.019	0.748 ± 0.237	0.617 ± 0.103	0.321 ± 0.068	0.438 ± 0.036	**0.041 ± 0.005**	**0.153 ± 0.012**
Koopa	0.360 ± 0.179	0.440 ± 0.076	0.434 ± 0.038	0.495 ± 0.023	0.357 ± 0.130	0.465 ± 0.077	0.048 ± 0.007	0.168 ± 0.016
OneNet	0.347 ± 0.049	0.478 ± 0.044	0.317 ± 0.059	0.477 ± 0.044	0.269 ± 0.029	0.423 ± 0.024	0.301 ± 0.011	0.460 ± 0.016
sKAF	0.324 ± 0.050	0.359 ± 0.019	16.68 ± 26.54	0.918 ± 0.644	0.572 ± 0.148	0.471 ± 0.037	0.190 ± 0.062	0.266 ± 0.013

	LuChenCheng		MooreSpiegel		NewtonLiepnik		NoseHoover	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	0.022 ± 0.000	0.110 ± 0.000	**0.102 ± 0.000**	**0.247 ± 0.000**	**0.138 ± 0.000**	**0.275 ± 0.000**	**0.116 ± 0.000**	**0.234 ± 0.000**
ModePlait	0.449 ± 0.024	0.524 ± 0.018	0.240 ± 0.023	0.387 ± 0.020	0.249 ± 0.026	0.390 ± 0.026	0.359 ± 0.008	0.468 ± 0.008
WPMixer	0.635 ± 0.846	0.486 ± 0.198	0.144 ± 0.011	0.264 ± 0.005	0.195 ± 0.014	0.352 ± 0.015	0.187 ± 0.010	0.312 ± 0.012
PAttn	0.350 ± 0.010	0.475 ± 0.008	0.137 ± 0.005	0.259 ± 0.009	0.250 ± 0.076	0.393 ± 0.058	0.223 ± 0.017	0.342 ± 0.016
Koopa	0.441 ± 0.028	0.506 ± 0.009	0.163 ± 0.016	0.289 ± 0.023	0.189 ± 0.004	0.346 ± 0.005	0.191 ± 0.023	0.320 ± 0.015
OneNet	0.434 ± 0.011	0.532 ± 0.007	0.252 ± 0.026	0.400 ± 0.024	0.235 ± 0.038	0.396 ± 0.038	0.317 ± 0.123	0.452 ± 0.088
sKAF	**0.013 ± 0.002**	**0.084 ± 0.005**	0.180 ± 0.021	0.269 ± 0.007	1.562 ± 0.231	0.496 ± 0.017	0.538 ± 0.439	0.417 ± 0.131

	NuclearQuadrupole		PehlivanWei		Qi		QiChen	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.052 ± 0.000**	**0.164 ± 0.000**	**0.069 ± 0.000**	**0.175 ± 0.000**	**0.191 ± 0.000**	**0.313 ± 0.000**	**0.096 ± 0.000**	**0.259 ± 0.000**
ModePlait	0.434 ± 0.025	0.536 ± 0.015	0.264 ± 0.018	0.383 ± 0.026	0.291 ± 0.043	0.435 ± 0.027	0.311 ± 0.007	0.464 ± 0.006
WPMixer	0.667 ± 0.655	0.569 ± 0.152	0.168 ± 0.017	0.304 ± 0.018	0.457 ± 0.169	0.536 ± 0.079	0.401 ± 0.303	0.484 ± 0.151
PAttn	0.495 ± 0.194	0.557 ± 0.089	0.282 ± 0.042	0.416 ± 0.037	0.578 ± 0.024	0.631 ± 0.015	0.735 ± 0.262	0.663 ± 0.129
Koopa	0.338 ± 0.018	0.467 ± 0.012	0.399 ± 0.340	0.421 ± 0.182	0.640 ± 0.070	0.591 ± 0.021	0.328 ± 0.026	0.450 ± 0.017
OneNet	0.474 ± 0.035	0.582 ± 0.026	0.216 ± 0.069	0.359 ± 0.058	0.281 ± 0.029	0.418 ± 0.010	0.310 ± 0.056	0.461 ± 0.050
sKAF	1.001 ± 0.297	0.632 ± 0.075	34.48 ± 12.68	1.805 ± 0.361	2.687 ± 1.281	0.663 ± 0.130	0.666 ± 0.432	0.382 ± 0.049

	RabinovichFabrikant		RayleighBenard		RikitakeDynamo		Rossler	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.098 ± 0.000**	**0.249 ± 0.000**	**0.155 ± 0.000**	**0.287 ± 0.000**	**0.227 ± 0.000**	**0.346 ± 0.000**	**0.018 ± 0.000**	**0.079 ± 0.000**
ModePlait	0.302 ± 0.024	0.444 ± 0.021	0.292 ± 0.024	0.439 ± 0.019	0.389 ± 0.038	0.509 ± 0.027	0.330 ± 0.054	0.424 ± 0.033
WPMixer	0.232 ± 0.126	0.361 ± 0.118	0.541 ± 0.608	0.510 ± 0.209	0.338 ± 0.172	0.405 ± 0.112	0.152 ± 0.019	0.282 ± 0.014
PAttn	0.389 ± 0.149	0.494 ± 0.118	0.325 ± 0.114	0.471 ± 0.086	0.335 ± 0.058	0.422 ± 0.065	0.265 ± 0.158	0.346 ± 0.100
Koopa	0.387 ± 0.125	0.498 ± 0.095	0.312 ± 0.037	0.443 ± 0.028	0.243 ± 0.019	0.361 ± 0.021	0.291 ± 0.122	0.365 ± 0.070
OneNet	0.495 ± 0.051	0.607 ± 0.032	0.300 ± 0.041	0.435 ± 0.032	0.394 ± 0.069	0.511 ± 0.050	0.431 ± 0.149	0.506 ± 0.088
sKAF	0.154 ± 0.038	0.270 ± 0.024	1.377 ± 0.355	0.660 ± 0.036	152.8 ± 103.3	4.673 ± 1.816	0.027 ± 0.014	0.094 ± 0.010

	Rucklidge		Sakarya		ShimizuMorioka		SprottA	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.134 ± 0.000**	**0.294 ± 0.000**	**0.064 ± 0.000**	**0.182 ± 0.000**	**0.155 ± 0.000**	**0.288 ± 0.000**	0.109 ± 0.000	0.241 ± 0.000
ModePlait	0.353 ± 0.030	0.492 ± 0.019	0.306 ± 0.023	0.431 ± 0.020	0.435 ± 0.017	0.558 ± 0.012	0.409 ± 0.031	0.516 ± 0.022
WPMixer	0.235 ± 0.030	0.385 ± 0.024	0.250 ± 0.116	0.383 ± 0.081	0.360 ± 0.007	0.492 ± 0.003	**0.086 ± 0.004**	**0.229 ± 0.005**
PAttn	0.947 ± 0.167	0.753 ± 0.062	0.260 ± 0.064	0.402 ± 0.049	0.507 ± 0.077	0.592 ± 0.050	0.091 ± 0.011	0.243 ± 0.016
Koopa	1.115 ± 0.512	0.746 ± 0.136	0.466 ± 0.318	0.489 ± 0.144	0.733 ± 0.324	0.667 ± 0.135	0.117 ± 0.018	0.272 ± 0.026
OneNet	0.390 ± 0.065	0.517 ± 0.048	0.334 ± 0.086	0.476 ± 0.060	0.382 ± 0.004	0.516 ± 0.003	0.371 ± 0.055	0.503 ± 0.039
sKAF	0.753 ± 0.217	0.481 ± 0.055	0.124 ± 0.042	0.240 ± 0.024	0.259 ± 0.064	0.341 ± 0.026	3.052 ± 0.966	0.821 ± 0.098

Table 10: Detailed multivariate forecasting results with forecasting step $\ell_s = 25$

	SprottB		SprottC		SprottD		SprottE	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.112 ± 0.000**	**0.240 ± 0.000**	**0.107 ± 0.000**	**0.215 ± 0.000**	**0.062 ± 0.000**	**0.186 ± 0.000**	**0.164 ± 0.000**	**0.288 ± 0.000**
ModePlait	0.300 ± 0.022	0.444 ± 0.018	0.242 ± 0.020	0.361 ± 0.015	0.210 ± 0.011	0.348 ± 0.010	0.293 ± 0.003	0.419 ± 0.008
WPMixer	0.854 ± 0.384	0.647 ± 0.139	0.232 ± 0.072	0.357 ± 0.050	0.168 ± 0.012	0.308 ± 0.011	0.755 ± 0.701	0.544 ± 0.205
PAttn	0.353 ± 0.040	0.474 ± 0.028	0.323 ± 0.116	0.417 ± 0.072	0.244 ± 0.072	0.365 ± 0.051	0.622 ± 0.228	0.600 ± 0.120
Koopa	0.555 ± 0.276	0.549 ± 0.102	0.396 ± 0.123	0.440 ± 0.078	0.274 ± 0.042	0.378 ± 0.020	0.806 ± 0.573	0.551 ± 0.151
OneNet	0.277 ± 0.015	0.412 ± 0.010	0.179 ± 0.007	0.302 ± 0.010	0.170 ± 0.002	0.327 ± 0.004	0.243 ± 0.030	0.387 ± 0.020
sKAF	2.280 ± 0.669	0.782 ± 0.116	1746 ± 1630	11.10 ± 7.170	218.1 ± 141.8	4.223 ± 1.191	2.362 ± 1.441	0.497 ± 0.100

	SprottF		SprottG		SprottH		SprottI	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.068 ± 0.000**	**0.194 ± 0.000**	0.010 ± 0.000	0.074 ± 0.000	**0.073 ± 0.000**	**0.196 ± 0.000**	0.023 ± 0.000	0.113 ± 0.000
ModePlait	0.264 ± 0.028	0.406 ± 0.022	0.327 ± 0.066	0.452 ± 0.045	0.273 ± 0.018	0.413 ± 0.014	0.469 ± 0.019	0.546 ± 0.025
WPMixer	0.228 ± 0.053	0.378 ± 0.040	1.147 ± 2.036	0.540 ± 0.394	0.223 ± 0.060	0.374 ± 0.043	0.168 ± 0.031	0.323 ± 0.024
PAttn	0.233 ± 0.054	0.382 ± 0.050	0.742 ± 0.118	0.585 ± 0.027	0.442 ± 0.137	0.524 ± 0.091	0.257 ± 0.102	0.405 ± 0.093
Koopa	0.551 ± 0.229	0.506 ± 0.054	0.183 ± 0.008	0.303 ± 0.012	0.225 ± 0.052	0.375 ± 0.038	0.628 ± 0.650	0.482 ± 0.251
OneNet	0.270 ± 0.061	0.422 ± 0.044	0.426 ± 0.108	0.498 ± 0.074	0.182 ± 0.004	0.349 ± 0.005	0.440 ± 0.078	0.548 ± 0.045
sKAF	1.707 ± 1.161	0.389 ± 0.107	**0.008 ± 0.001**	**0.055 ± 0.002**	0.729 ± 0.260	0.331 ± 0.049	**0.023 ± 0.002**	**0.094 ± 0.004**

	SprottJ		SprottJerk		SprottK		SprottL	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.013 ± 0.000**	**0.082 ± 0.000**	**0.037 ± 0.000**	0.136 ± 0.000	0.018 ± 0.000	0.089 ± 0.000	0.071 ± 0.000	0.203 ± 0.000
ModePlait	0.459 ± 0.045	0.547 ± 0.033	0.489 ± 0.069	0.558 ± 0.041	0.440 ± 0.047	0.525 ± 0.033	0.311 ± 0.052	0.448 ± 0.036
WPMixer	0.180 ± 0.012	0.344 ± 0.015	0.271 ± 0.315	0.312 ± 0.188	0.244 ± 0.006	0.389 ± 0.006	0.128 ± 0.011	0.270 ± 0.020
PAttn	0.470 ± 0.128	0.549 ± 0.086	0.090 ± 0.043	0.225 ± 0.064	0.227 ± 0.009	0.374 ± 0.012	0.153 ± 0.024	0.305 ± 0.037
Koopa	0.645 ± 0.725	0.584 ± 0.266	0.071 ± 0.028	0.203 ± 0.044	0.279 ± 0.106	0.401 ± 0.062	0.328 ± 0.165	0.413 ± 0.081
OneNet	0.501 ± 0.067	0.586 ± 0.031	0.607 ± 0.053	0.652 ± 0.034	0.485 ± 0.010	0.573 ± 0.010	0.466 ± 0.040	0.571 ± 0.027
sKAF	0.215 ± 0.069	0.179 ± 0.015	0.058 ± 0.009	**0.116 ± 0.007**	**0.015 ± 0.001**	**0.086 ± 0.001**	**0.055 ± 0.010**	**0.159 ± 0.009**

	SprottM		SprottN		SprottO		SprottP	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.012 ± 0.000**	**0.079 ± 0.000**	**0.013 ± 0.000**	**0.090 ± 0.000**	0.017 ± 0.000	0.105 ± 0.000	**0.100 ± 0.000**	**0.222 ± 0.000**
ModePlait	0.376 ± 0.006	0.496 ± 0.005	0.385 ± 0.018	0.496 ± 0.012	0.592 ± 0.031	0.584 ± 0.024	0.265 ± 0.031	0.405 ± 0.024
WPMixer	3.811 ± 2.347	1.268 ± 0.492	0.358 ± 0.191	0.448 ± 0.094	1.718 ± 3.766	0.384 ± 0.530	0.436 ± 0.132	0.486 ± 0.047
PAttn	0.596 ± 0.264	0.610 ± 0.136	0.296 ± 0.028	0.424 ± 0.015	0.029 ± 0.004	0.134 ± 0.007	0.471 ± 0.035	0.507 ± 0.020
Koopa	1.320 ± 1.205	0.799 ± 0.267	0.343 ± 0.024	0.474 ± 0.025	0.054 ± 0.030	0.178 ± 0.049	0.463 ± 0.034	0.494 ± 0.016
OneNet	0.540 ± 0.004	0.597 ± 0.003	0.524 ± 0.004	0.596 ± 0.005	0.819 ± 0.068	0.797 ± 0.033	0.364 ± 0.016	0.495 ± 0.013
sKAF	0.037 ± 0.007	0.112 ± 0.006	0.024 ± 0.007	0.095 ± 0.010	**0.005 ± 0.000**	**0.054 ± 0.001**	1.639 ± 0.636	0.556 ± 0.089

	SprottQ		SprottR		SprottS		SprottTorus	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.052 ± 0.000**	**0.141 ± 0.000**	**0.043 ± 0.000**	**0.124 ± 0.000**	**0.027 ± 0.000**	**0.127 ± 0.000**	**0.101 ± 0.000**	**0.184 ± 0.000**
ModePlait	0.427 ± 0.028	0.509 ± 0.015	0.232 ± 0.010	0.357 ± 0.009	0.446 ± 0.073	0.530 ± 0.054	0.219 ± 0.007	0.338 ± 0.010
WPMixer	0.434 ± 0.459	0.476 ± 0.177	0.274 ± 0.051	0.370 ± 0.041	0.657 ± 0.462	0.539 ± 0.175	0.328 ± 0.125	0.412 ± 0.085
PAttn	0.369 ± 0.039	0.461 ± 0.035	0.324 ± 0.034	0.407 ± 0.025	0.449 ± 0.158	0.499 ± 0.084	0.398 ± 0.139	0.455 ± 0.094
Koopa	0.673 ± 0.587	0.578 ± 0.229	0.489 ± 0.272	0.463 ± 0.099	0.541 ± 0.350	0.547 ± 0.143	0.338 ± 0.083	0.430 ± 0.032
OneNet	0.484 ± 0.004	0.538 ± 0.003	0.267 ± 0.039	0.403 ± 0.024	0.461 ± 0.074	0.561 ± 0.047	0.191 ± 0.006	0.295 ± 0.005
sKAF	0.552 ± 0.176	0.371 ± 0.052	1.806 ± 2.112	0.284 ± 0.104	3.377 ± 2.446	0.864 ± 0.184	10.06 ± 3.954	0.520 ± 0.076

	VallisElNino		WangSun		ZhouChen	
	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	0.205 ± 0.000	**0.343 ± 0.000**	0.095 ± 0.000	0.221 ± 0.000	**0.070 ± 0.000**	**0.175 ± 0.000**
ModePlait	**0.196 ± 0.014**	0.355 ± 0.009	0.102 ± 0.009	0.228 ± 0.012	0.140 ± 0.024	0.236 ± 0.022
WPMixer	0.206 ± 0.025	0.361 ± 0.014	0.139 ± 0.087	0.243 ± 0.046	1.328 ± 2.633	0.290 ± 0.123
PAttn	0.244 ± 0.004	0.394 ± 0.004	0.137 ± 0.010	0.271 ± 0.013	0.259 ± 0.102	0.319 ± 0.069
Koopa	0.270 ± 0.040	0.414 ± 0.024	0.170 ± 0.117	0.263 ± 0.058	0.289 ± 0.162	0.275 ± 0.042
OneNet	0.233 ± 0.009	0.387 ± 0.009	**0.076 ± 0.006**	**0.195 ± 0.010**	0.155 ± 0.080	0.230 ± 0.034
sKAF	0.220 ± 0.009	0.357 ± 0.003	331.4 ± 192.8	4.178 ± 1.024	219.9 ± 183.8	3.878 ± 1.574

Table 11: Detailed multivariate forecasting results with forecasting step $\ell_s = 30$

	Aizawa		Arneodo		Bouali2		BurkeShaw	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.039 ± 0.000**	**0.136 ± 0.000**	**0.044 ± 0.000**	**0.141 ± 0.000**	**0.100 ± 0.000**	**0.213 ± 0.000**	**0.180 ± 0.000**	**0.309 ± 0.000**
ModePlait	0.395 ± 0.018	0.489 ± 0.010	0.369 ± 0.025	0.463 ± 0.017	0.162 ± 0.005	0.306 ± 0.002	0.244 ± 0.004	0.384 ± 0.003
WPMixer	3.188 ± 5.154	0.845 ± 0.453	0.351 ± 0.375	0.350 ± 0.099	0.129 ± 0.009	0.244 ± 0.006	0.228 ± 0.110	0.370 ± 0.063
PAttn	0.543 ± 0.309	0.560 ± 0.143	0.168 ± 0.013	0.289 ± 0.012	0.194 ± 0.057	0.291 ± 0.034	0.312 ± 0.077	0.445 ± 0.048
Koopa	0.887 ± 0.137	0.699 ± 0.043	0.167 ± 0.012	0.297 ± 0.017	0.161 ± 0.018	0.276 ± 0.016	0.447 ± 0.239	0.489 ± 0.136
OneNet	0.431 ± 0.011	0.509 ± 0.005	0.509 ± 0.024	0.585 ± 0.010	0.122 ± 0.007	0.268 ± 0.008	0.237 ± 0.002	0.388 ± 0.002
sKAF	0.612 ± 0.252	0.378 ± 0.064	0.518 ± 0.532	0.269 ± 0.054	185.1 ± 301.0	2.884 ± 1.269	15.85 ± 10.97	1.061 ± 0.252

	Chen		ChenLee		Dadras		DequanLi	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.237 ± 0.000**	**0.382 ± 0.000**	**0.195 ± 0.000**	**0.287 ± 0.000**	**0.095 ± 0.000**	**0.228 ± 0.000**	**0.326 ± 0.000**	0.412 ± 0.000
ModePlait	0.363 ± 0.017	0.472 ± 0.009	0.280 ± 0.016	0.398 ± 0.015	0.157 ± 0.019	0.312 ± 0.022	0.341 ± 0.042	0.454 ± 0.028
WPMixer	0.389 ± 0.129	0.506 ± 0.080	0.206 ± 0.017	0.320 ± 0.019	0.105 ± 0.005	0.248 ± 0.005	1.644 ± 1.550	0.703 ± 0.149
PAttn	0.402 ± 0.058	0.517 ± 0.038	0.294 ± 0.170	0.389 ± 0.124	0.131 ± 0.006	0.274 ± 0.008	0.459 ± 0.328	**0.389 ± 0.076**
Koopa	0.699 ± 0.095	0.616 ± 0.032	0.203 ± 0.008	0.329 ± 0.014	0.121 ± 0.010	0.267 ± 0.014	0.515 ± 0.045	0.553 ± 0.037
OneNet	0.339 ± 0.064	0.476 ± 0.047	0.272 ± 0.021	0.340 ± 0.038	0.138 ± 0.027	0.295 ± 0.031	0.342 ± 0.045	0.479 ± 0.032
sKAF	0.667 ± 0.383	0.490 ± 0.081	22.27 ± 10.25	1.399 ± 0.361	17.69 ± 11.97	1.133 ± 0.226	91.31 ± 14.72	6.631 ± 0.533

	Finance		GenesioTesi		GuckenheimerHolmes		Hadley	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.195 ± 0.000**	**0.342 ± 0.000**	**0.012 ± 0.000**	**0.087 ± 0.000**	**0.058 ± 0.000**	**0.146 ± 0.000**	**0.046 ± 0.000**	**0.162 ± 0.000**
ModePlait	0.286 ± 0.031	0.435 ± 0.017	0.344 ± 0.010	0.449 ± 0.006	0.143 ± 0.029	0.247 ± 0.024	0.302 ± 0.032	0.416 ± 0.016
WPMixer	0.209 ± 0.016	0.354 ± 0.005	0.246 ± 0.032	0.407 ± 0.026	0.119 ± 0.018	0.210 ± 0.016	0.538 ± 0.292	0.505 ± 0.077
PAttn	0.273 ± 0.022	0.390 ± 0.021	0.327 ± 0.053	0.457 ± 0.058	0.188 ± 0.057	0.261 ± 0.014	0.270 ± 0.020	0.423 ± 0.013
Koopa	0.240 ± 0.022	0.381 ± 0.020	0.513 ± 0.320	0.505 ± 0.099	0.308 ± 0.121	0.308 ± 0.043	0.441 ± 0.140	0.511 ± 0.070
OneNet	0.289 ± 0.007	0.435 ± 0.006	0.411 ± 0.104	0.529 ± 0.067	0.183 ± 0.016	0.275 ± 0.007	0.244 ± 0.042	0.386 ± 0.023
sKAF	0.936 ± 0.182	0.540 ± 0.032	0.084 ± 0.031	0.205 ± 0.038	302.0 ± 154.4	7.272 ± 1.994	3.610 ± 0.868	0.834 ± 0.093

	Halvorsen		HenonHeiles		HyperBao		HyperCai	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.120 ± 0.000**	**0.257 ± 0.000**	**0.028 ± 0.000**	**0.124 ± 0.000**	**0.132 ± 0.000**	**0.275 ± 0.000**	**0.275 ± 0.000**	**0.371 ± 0.000**
ModePlait	0.400 ± 0.038	0.509 ± 0.025	0.538 ± 0.052	0.594 ± 0.030	0.255 ± 0.023	0.400 ± 0.018	0.358 ± 0.006	0.476 ± 0.004
WPMixer	0.387 ± 0.397	0.422 ± 0.184	0.137 ± 0.032	0.306 ± 0.035	0.343 ± 0.165	0.440 ± 0.088	0.506 ± 0.166	0.543 ± 0.078
PAttn	0.168 ± 0.020	0.329 ± 0.022	0.129 ± 0.012	0.292 ± 0.014	0.273 ± 0.037	0.414 ± 0.035	0.414 ± 0.120	0.501 ± 0.079
Koopa	0.244 ± 0.056	0.387 ± 0.043	0.150 ± 0.031	0.307 ± 0.032	0.252 ± 0.029	0.394 ± 0.021	0.348 ± 0.149	0.461 ± 0.096
OneNet	0.376 ± 0.024	0.504 ± 0.018	0.623 ± 0.061	0.678 ± 0.037	0.188 ± 0.003	0.354 ± 0.003	0.326 ± 0.050	0.470 ± 0.033
sKAF	0.383 ± 0.086	0.344 ± 0.016	3.324 ± 0.539	1.020 ± 0.069	15.50 ± 6.017	1.478 ± 0.272	52.62 ± 39.40	3.850 ± 0.792

	HyperJha		HyperLorenz		HyperLu		HyperPang	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.067 ± 0.000**	**0.190 ± 0.000**	**0.153 ± 0.000**	**0.291 ± 0.000**	**0.086 ± 0.000**	**0.217 ± 0.000**	**0.162 ± 0.000**	**0.309 ± 0.000**
ModePlait	0.324 ± 0.062	0.479 ± 0.042	0.280 ± 0.019	0.431 ± 0.015	0.233 ± 0.007	0.391 ± 0.006	0.256 ± 0.032	0.401 ± 0.027
WPMixer	1.063 ± 1.374	0.626 ± 0.225	0.269 ± 0.054	0.394 ± 0.024	0.244 ± 0.159	0.375 ± 0.091	0.401 ± 0.255	0.468 ± 0.123
PAttn	0.335 ± 0.123	0.456 ± 0.082	0.308 ± 0.008	0.413 ± 0.006	0.312 ± 0.195	0.421 ± 0.090	0.262 ± 0.049	0.414 ± 0.038
Koopa	0.318 ± 0.026	0.465 ± 0.019	0.310 ± 0.011	0.430 ± 0.006	0.330 ± 0.191	0.437 ± 0.114	0.233 ± 0.032	0.399 ± 0.031
OneNet	0.280 ± 0.056	0.440 ± 0.046	0.300 ± 0.049	0.457 ± 0.031	0.169 ± 0.021	0.335 ± 0.026	0.208 ± 0.043	0.367 ± 0.035
sKAF	1.418 ± 0.551	0.715 ± 0.104	7.582 ± 1.891	1.134 ± 0.106	6.991 ± 1.830	1.082 ± 0.132	98.26 ± 53.18	3.211 ± 0.715

	HyperQi		HyperRossler		HyperWang		HyperXu	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.145 ± 0.000**	**0.237 ± 0.000**	**0.073 ± 0.000**	0.180 ± 0.000	0.259 ± 0.000	0.379 ± 0.000	0.223 ± 0.000	0.376 ± 0.000
ModePlait	0.183 ± 0.013	0.305 ± 0.015	0.097 ± 0.016	0.196 ± 0.017	0.257 ± 0.039	0.383 ± 0.031	0.260 ± 0.033	0.400 ± 0.024
WPMixer	0.489 ± 0.433	0.377 ± 0.117	0.139 ± 0.037	0.206 ± 0.028	0.237 ± 0.012	0.391 ± 0.010	0.252 ± 0.067	0.389 ± 0.038
PAttn	0.289 ± 0.156	0.348 ± 0.053	0.096 ± 0.006	**0.179 ± 0.010**	0.372 ± 0.090	0.481 ± 0.056	0.294 ± 0.009	0.433 ± 0.007
Koopa	0.212 ± 0.075	0.326 ± 0.043	0.109 ± 0.036	0.199 ± 0.030	0.570 ± 0.378	0.496 ± 0.121	0.262 ± 0.010	0.405 ± 0.007
OneNet	0.168 ± 0.037	0.288 ± 0.040	0.125 ± 0.023	0.234 ± 0.038	**0.185 ± 0.029**	**0.342 ± 0.024**	**0.212 ± 0.020**	**0.372 ± 0.016**
sKAF	3.553 ± 1.947	0.803 ± 0.215	652.4 ± 292.2	9.395 ± 2.691	299.2 ± 89.98	8.367 ± 1.223	74.07 ± 84.62	3.587 ± 1.666

Table 12: Detailed multivariate forecasting results with forecasting step $\ell_s = 30$

	HyperYan		HyperYangChen		KawczynskiStrizhak		Laser	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.144 ± 0.000**	**0.289 ± 0.000**	**0.215 ± 0.000**	**0.357 ± 0.000**	**0.059 ± 0.000**	0.132 ± 0.000	**0.179 ± 0.000**	**0.304 ± 0.000**
ModePlait	0.259 ± 0.011	0.395 ± 0.007	0.334 ± 0.010	0.478 ± 0.011	0.064 ± 0.000	0.143 ± 0.002	0.371 ± 0.009	0.495 ± 0.008
WPMixer	0.261 ± 0.068	0.394 ± 0.050	0.286 ± 0.041	0.435 ± 0.030	0.065 ± 0.002	**0.131 ± 0.002**	0.513 ± 0.425	0.520 ± 0.174
PAttn	0.327 ± 0.120	0.445 ± 0.090	0.407 ± 0.077	0.520 ± 0.034	0.070 ± 0.003	0.135 ± 0.003	0.414 ± 0.108	0.512 ± 0.067
Koopa	0.297 ± 0.038	0.417 ± 0.029	0.382 ± 0.024	0.502 ± 0.014	0.071 ± 0.004	0.137 ± 0.006	0.339 ± 0.150	0.454 ± 0.066
OneNet	0.236 ± 0.065	0.377 ± 0.058	0.260 ± 0.022	0.421 ± 0.021	0.069 ± 0.001	0.152 ± 0.002	0.385 ± 0.003	0.501 ± 0.003
sKAF	5.138 ± 4.698	0.709 ± 0.194	13.64 ± 14.42	1.597 ± 0.539	0.111 ± 0.015	0.190 ± 0.009	76.07 ± 12.76	5.500 ± 0.551

	Lorenz		LorenzBounded		LorenzStenflo		LuChen	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.194 ± 0.000**	**0.339 ± 0.000**	**0.137 ± 0.000**	**0.272 ± 0.000**	0.281 ± 0.000	0.421 ± 0.000	0.141 ± 0.000	0.307 ± 0.000
ModePlait	0.323 ± 0.030	0.459 ± 0.023	0.399 ± 0.011	0.522 ± 0.008	0.321 ± 0.020	0.462 ± 0.015	0.255 ± 0.025	0.406 ± 0.019
WPMixer	0.385 ± 0.173	0.468 ± 0.077	0.386 ± 0.168	0.463 ± 0.067	0.286 ± 0.062	0.417 ± 0.043	0.056 ± 0.011	0.183 ± 0.022
PAttn	0.278 ± 0.016	0.413 ± 0.013	0.679 ± 0.179	0.595 ± 0.081	0.315 ± 0.064	0.425 ± 0.038	**0.048 ± 0.007**	**0.166 ± 0.013**
Koopa	0.369 ± 0.172	0.450 ± 0.063	0.441 ± 0.041	0.492 ± 0.021	0.331 ± 0.082	0.448 ± 0.056	0.073 ± 0.012	0.211 ± 0.015
OneNet	0.320 ± 0.053	0.457 ± 0.040	0.344 ± 0.040	0.496 ± 0.030	**0.253 ± 0.028**	**0.410 ± 0.021**	0.287 ± 0.008	0.454 ± 0.011
sKAF	0.393 ± 0.115	0.393 ± 0.035	1335 ± 925.8	8.336 ± 2.552	0.466 ± 0.016	0.434 ± 0.005	0.213 ± 0.079	0.294 ± 0.010

	LuChenCheng		MooreSpiegel		NewtonLiepnik		NoseHoover	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	0.034 ± 0.000	0.140 ± 0.000	**0.116 ± 0.000**	0.261 ± 0.000	**0.152 ± 0.000**	**0.296 ± 0.000**	**0.117 ± 0.000**	**0.239 ± 0.000**
ModePlait	0.490 ± 0.015	0.553 ± 0.013	0.272 ± 0.028	0.412 ± 0.022	0.261 ± 0.020	0.397 ± 0.022	0.382 ± 0.013	0.489 ± 0.008
WPMixer	0.452 ± 0.433	0.466 ± 0.145	0.148 ± 0.013	0.271 ± 0.013	0.193 ± 0.013	0.349 ± 0.012	0.152 ± 0.015	0.282 ± 0.016
PAttn	0.352 ± 0.023	0.481 ± 0.025	0.145 ± 0.009	0.262 ± 0.009	0.263 ± 0.068	0.405 ± 0.056	0.209 ± 0.022	0.326 ± 0.021
Koopa	0.357 ± 0.026	0.462 ± 0.014	0.157 ± 0.006	0.288 ± 0.011	0.185 ± 0.005	0.343 ± 0.005	0.215 ± 0.029	0.334 ± 0.024
OneNet	0.434 ± 0.013	0.530 ± 0.007	0.259 ± 0.020	0.403 ± 0.019	0.227 ± 0.033	0.386 ± 0.030	0.310 ± 0.120	0.446 ± 0.089
sKAF	**0.019 ± 0.002**	**0.102 ± 0.003**	0.163 ± 0.019	**0.256 ± 0.003**	1.621 ± 0.168	0.550 ± 0.016	0.904 ± 0.340	0.551 ± 0.097

	NuclearQuadrupole		PehlivanWei		Qi		QiChen	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.077 ± 0.000**	**0.201 ± 0.000**	**0.074 ± 0.000**	**0.188 ± 0.000**	**0.240 ± 0.000**	**0.359 ± 0.000**	**0.118 ± 0.000**	**0.287 ± 0.000**
ModePlait	0.452 ± 0.030	0.552 ± 0.014	0.284 ± 0.019	0.399 ± 0.023	0.304 ± 0.049	0.445 ± 0.030	0.342 ± 0.007	0.485 ± 0.009
WPMixer	0.551 ± 0.412	0.545 ± 0.106	0.166 ± 0.021	0.299 ± 0.023	0.436 ± 0.144	0.535 ± 0.072	0.348 ± 0.181	0.460 ± 0.099
PAttn	0.494 ± 0.140	0.556 ± 0.064	0.283 ± 0.043	0.415 ± 0.035	0.552 ± 0.042	0.621 ± 0.025	0.660 ± 0.196	0.627 ± 0.098
Koopa	0.300 ± 0.015	0.435 ± 0.011	0.306 ± 0.219	0.383 ± 0.135	0.566 ± 0.084	0.587 ± 0.034	0.327 ± 0.020	0.454 ± 0.011
OneNet	0.485 ± 0.026	0.587 ± 0.022	0.214 ± 0.077	0.357 ± 0.066	0.281 ± 0.034	0.419 ± 0.013	0.329 ± 0.048	0.476 ± 0.041
sKAF	0.723 ± 0.154	0.584 ± 0.053	58.62 ± 29.91	2.354 ± 0.436	4.466 ± 1.975	0.934 ± 0.102	0.407 ± 0.154	0.351 ± 0.023

	RabinovichFabrikant		RayleighBenard		RikitakeDynamo		Rossler	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.120 ± 0.000**	**0.275 ± 0.000**	**0.153 ± 0.000**	**0.297 ± 0.000**	**0.263 ± 0.000**	0.376 ± 0.000	0.020 ± 0.000	0.084 ± 0.000
ModePlait	0.328 ± 0.024	0.463 ± 0.018	0.309 ± 0.025	0.450 ± 0.022	0.413 ± 0.042	0.530 ± 0.027	0.360 ± 0.054	0.445 ± 0.034
WPMixer	0.227 ± 0.123	0.356 ± 0.118	0.466 ± 0.486	0.484 ± 0.195	0.321 ± 0.135	0.398 ± 0.101	0.159 ± 0.027	0.290 ± 0.023
PAttn	0.434 ± 0.137	0.523 ± 0.098	0.341 ± 0.135	0.468 ± 0.093	0.336 ± 0.070	0.422 ± 0.073	0.283 ± 0.114	0.370 ± 0.075
Koopa	0.419 ± 0.113	0.508 ± 0.071	0.295 ± 0.016	0.426 ± 0.007	0.272 ± 0.018	**0.375 ± 0.035**	0.266 ± 0.091	0.354 ± 0.051
OneNet	0.499 ± 0.045	0.610 ± 0.031	0.299 ± 0.044	0.435 ± 0.034	0.418 ± 0.077	0.531 ± 0.062	0.431 ± 0.149	0.502 ± 0.093
sKAF	0.216 ± 0.057	0.327 ± 0.034	2.231 ± 0.863	0.756 ± 0.060	251.6 ± 352.9	5.020 ± 3.860	**0.016 ± 0.006**	**0.076 ± 0.007**

	Rucklidge		Sakarya		ShimizuMorioka		SprottA	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.150 ± 0.000**	**0.307 ± 0.000**	**0.078 ± 0.000**	**0.203 ± 0.000**	**0.193 ± 0.000**	**0.326 ± 0.000**	0.135 ± 0.000	0.274 ± 0.000
ModePlait	0.375 ± 0.033	0.503 ± 0.020	0.327 ± 0.025	0.442 ± 0.022	0.453 ± 0.016	0.569 ± 0.012	0.441 ± 0.043	0.536 ± 0.027
WPMixer	0.234 ± 0.033	0.386 ± 0.025	0.275 ± 0.203	0.380 ± 0.100	0.369 ± 0.009	0.505 ± 0.010	0.090 ± 0.009	0.237 ± 0.010
PAttn	0.891 ± 0.194	0.728 ± 0.068	0.234 ± 0.062	0.382 ± 0.050	0.529 ± 0.048	0.603 ± 0.025	**0.082 ± 0.004**	**0.229 ± 0.010**
Koopa	0.976 ± 0.445	0.686 ± 0.149	0.311 ± 0.127	0.438 ± 0.097	0.640 ± 0.120	0.621 ± 0.062	0.119 ± 0.015	0.268 ± 0.018
OneNet	0.399 ± 0.044	0.523 ± 0.034	0.320 ± 0.088	0.461 ± 0.065	0.374 ± 0.007	0.512 ± 0.003	0.367 ± 0.044	0.502 ± 0.031
sKAF	0.923 ± 0.472	0.526 ± 0.117	0.139 ± 0.008	0.261 ± 0.005	0.366 ± 0.102	0.404 ± 0.033	3.561 ± 1.530	0.897 ± 0.155

Table 13: Detailed multivariate forecasting results with forecasting step $\ell_s = 30$

	SprottB		SprottC		SprottD		SprottE	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.123 ± 0.000**	**0.255 ± 0.000**	**0.141 ± 0.000**	**0.252 ± 0.000**	**0.083 ± 0.000**	**0.221 ± 0.000**	**0.186 ± 0.000**	**0.318 ± 0.000**
ModePlait	0.314 ± 0.024	0.454 ± 0.014	0.254 ± 0.019	0.371 ± 0.015	0.235 ± 0.015	0.366 ± 0.010	0.282 ± 0.004	0.416 ± 0.005
WPMixer	0.675 ± 0.293	0.610 ± 0.123	0.245 ± 0.081	0.363 ± 0.050	0.195 ± 0.015	0.327 ± 0.010	0.617 ± 0.519	0.527 ± 0.171
PAttn	0.337 ± 0.044	0.472 ± 0.026	0.352 ± 0.134	0.436 ± 0.078	0.270 ± 0.065	0.389 ± 0.044	0.543 ± 0.175	0.570 ± 0.105
Koopa	0.544 ± 0.257	0.544 ± 0.100	0.429 ± 0.117	0.456 ± 0.071	0.286 ± 0.043	0.388 ± 0.018	0.595 ± 0.316	0.521 ± 0.108
OneNet	0.272 ± 0.014	0.409 ± 0.008	0.183 ± 0.005	0.302 ± 0.008	0.179 ± 0.001	0.335 ± 0.002	0.239 ± 0.031	0.381 ± 0.026
sKAF	1.690 ± 0.473	0.665 ± 0.051	1005 ± 712.8	9.608 ± 3.717	110.7 ± 68.11	3.802 ± 1.081	1.700 ± 0.968	0.495 ± 0.051

	SprottF		SprottG		SprottH		SprottI	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.109 ± 0.000**	**0.254 ± 0.000**	0.013 ± 0.000	0.086 ± 0.000	**0.132 ± 0.000**	**0.270 ± 0.000**	**0.030 ± 0.000**	0.131 ± 0.000
ModePlait	0.311 ± 0.032	0.441 ± 0.026	0.345 ± 0.067	0.462 ± 0.046	0.299 ± 0.019	0.436 ± 0.016	0.502 ± 0.007	0.569 ± 0.008
WPMixer	0.220 ± 0.047	0.375 ± 0.037	1.161 ± 2.068	0.528 ± 0.380	0.228 ± 0.058	0.380 ± 0.037	0.179 ± 0.026	0.333 ± 0.026
PAttn	0.248 ± 0.079	0.391 ± 0.065	0.828 ± 0.175	0.614 ± 0.050	0.483 ± 0.176	0.538 ± 0.111	0.284 ± 0.135	0.422 ± 0.111
Koopa	0.367 ± 0.082	0.459 ± 0.038	0.173 ± 0.013	0.296 ± 0.010	0.230 ± 0.063	0.376 ± 0.041	0.473 ± 0.417	0.450 ± 0.192
OneNet	0.276 ± 0.050	0.429 ± 0.032	0.428 ± 0.106	0.503 ± 0.073	0.186 ± 0.003	0.352 ± 0.004	0.431 ± 0.081	0.542 ± 0.046
sKAF	2.665 ± 2.091	0.514 ± 0.170	**0.008 ± 0.001**	**0.056 ± 0.002**	0.930 ± 0.577	0.358 ± 0.072	0.046 ± 0.013	**0.124 ± 0.011**

	SprottJ		SprottJerk		SprottK		SprottL	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.017 ± 0.000**	**0.100 ± 0.000**	**0.049 ± 0.000**	0.161 ± 0.000	0.021 ± 0.000	0.100 ± 0.000	0.089 ± 0.000	0.233 ± 0.000
ModePlait	0.526 ± 0.036	0.590 ± 0.028	0.522 ± 0.075	0.582 ± 0.040	0.494 ± 0.058	0.562 ± 0.038	0.338 ± 0.053	0.465 ± 0.034
WPMixer	0.201 ± 0.008	0.365 ± 0.010	0.227 ± 0.255	0.302 ± 0.162	0.225 ± 0.022	0.375 ± 0.021	0.134 ± 0.014	0.280 ± 0.022
PAttn	0.501 ± 0.115	0.569 ± 0.070	0.097 ± 0.049	0.233 ± 0.073	0.220 ± 0.026	0.373 ± 0.025	0.171 ± 0.016	0.311 ± 0.025
Koopa	1.133 ± 1.773	0.633 ± 0.353	0.081 ± 0.025	0.223 ± 0.041	0.258 ± 0.047	0.394 ± 0.037	0.298 ± 0.117	0.405 ± 0.072
OneNet	0.509 ± 0.063	0.593 ± 0.029	0.609 ± 0.028	0.654 ± 0.018	0.490 ± 0.010	0.574 ± 0.010	0.425 ± 0.066	0.541 ± 0.047
sKAF	0.259 ± 0.082	0.225 ± 0.021	0.081 ± 0.019	**0.134 ± 0.009**	**0.019 ± 0.001**	**0.090 ± 0.003**	**0.087 ± 0.047**	**0.186 ± 0.021**

	SprottM		SprottN		SprottO		SprottP	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.018 ± 0.000**	**0.099 ± 0.000**	**0.020 ± 0.000**	**0.112 ± 0.000**	0.025 ± 0.000	0.129 ± 0.000	**0.116 ± 0.000**	**0.255 ± 0.000**
ModePlait	0.419 ± 0.009	0.529 ± 0.006	0.436 ± 0.015	0.530 ± 0.009	0.641 ± 0.030	0.618 ± 0.018	0.299 ± 0.032	0.438 ± 0.022
WPMixer	3.323 ± 1.775	1.195 ± 0.413	0.347 ± 0.139	0.454 ± 0.066	1.083 ± 2.332	0.362 ± 0.447	0.473 ± 0.120	0.512 ± 0.035
PAttn	0.624 ± 0.210	0.638 ± 0.105	0.300 ± 0.027	0.429 ± 0.021	0.025 ± 0.005	0.128 ± 0.013	0.547 ± 0.056	0.544 ± 0.030
Koopa	1.103 ± 0.916	0.772 ± 0.248	0.364 ± 0.013	0.481 ± 0.033	0.070 ± 0.030	0.206 ± 0.050	0.559 ± 0.062	0.537 ± 0.029
OneNet	0.552 ± 0.009	0.603 ± 0.006	0.529 ± 0.006	0.598 ± 0.005	0.797 ± 0.055	0.786 ± 0.026	0.380 ± 0.013	0.510 ± 0.011
sKAF	0.044 ± 0.012	0.124 ± 0.010	0.048 ± 0.017	0.116 ± 0.007	**0.006 ± 0.000**	**0.058 ± 0.001**	4.961 ± 3.451	0.851 ± 0.287

	SprottQ		SprottR		SprottS		SprottTorus	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.067 ± 0.000**	**0.161 ± 0.000**	**0.056 ± 0.000**	**0.153 ± 0.000**	**0.037 ± 0.000**	**0.148 ± 0.000**	**0.123 ± 0.000**	**0.209 ± 0.000**
ModePlait	0.465 ± 0.035	0.535 ± 0.023	0.263 ± 0.009	0.383 ± 0.010	0.495 ± 0.087	0.564 ± 0.059	0.222 ± 0.011	0.341 ± 0.014
WPMixer	0.468 ± 0.484	0.498 ± 0.179	0.233 ± 0.029	0.354 ± 0.026	0.899 ± 0.734	0.592 ± 0.222	0.314 ± 0.078	0.413 ± 0.068
PAttn	0.393 ± 0.025	0.479 ± 0.029	0.298 ± 0.022	0.395 ± 0.010	0.434 ± 0.173	0.494 ± 0.080	0.405 ± 0.131	0.465 ± 0.090
Koopa	0.718 ± 0.737	0.578 ± 0.228	0.345 ± 0.155	0.409 ± 0.070	0.487 ± 0.187	0.538 ± 0.107	0.360 ± 0.056	0.444 ± 0.025
OneNet	0.489 ± 0.002	0.542 ± 0.003	0.270 ± 0.046	0.407 ± 0.028	0.482 ± 0.067	0.576 ± 0.038	0.200 ± 0.003	0.300 ± 0.003
sKAF	1.262 ± 0.612	0.502 ± 0.086	2.698 ± 1.692	0.372 ± 0.102	9.455 ± 10.42	1.253 ± 0.485	12.67 ± 3.970	0.648 ± 0.104

	VallisElNino		WangSun		ZhouChen	
	MSE	MAE	MSE	MAE	MSE	MAE
AdaKoop	**0.197 ± 0.000**	**0.343 ± 0.000**	0.101 ± 0.000	0.234 ± 0.000	**0.107 ± 0.000**	**0.204 ± 0.000**
ModePlait	0.211 ± 0.011	0.369 ± 0.011	0.109 ± 0.008	0.238 ± 0.010	0.154 ± 0.027	0.247 ± 0.025
WPMixer	0.207 ± 0.024	0.359 ± 0.011	0.126 ± 0.058	0.241 ± 0.037	0.523 ± 0.785	0.300 ± 0.115
PAttn	0.235 ± 0.010	0.384 ± 0.011	0.144 ± 0.004	0.276 ± 0.006	0.282 ± 0.105	0.339 ± 0.075
Koopa	0.238 ± 0.008	0.389 ± 0.006	0.172 ± 0.101	0.260 ± 0.049	0.270 ± 0.128	0.289 ± 0.036
OneNet	0.232 ± 0.009	0.383 ± 0.011	**0.079 ± 0.008**	**0.199 ± 0.014**	0.160 ± 0.076	0.237 ± 0.035
sKAF	0.273 ± 0.014	0.396 ± 0.004	427.2 ± 285.0	4.738 ± 1.181	582.9 ± 407.5	7.425 ± 2.647