Discovering Dynamics with Kolmogorov–Arnold Networks: Linear Multistep Method-Based Algorithms and Error Estimation

Jintao Hu¹, Hongjiong Tian¹, and Qian Guo^{1,*}

¹Department of Mathematics, Shanghai Normal University, P. R. China

Abstract

Uncovering the underlying dynamics from observed data is a critical task in various scientific fields. Recent advances have shown that combining deep learning techniques with linear multistep methods (LMMs) can be highly effective for this purpose. In this work, we propose a novel framework that integrates Kolmogorov–Arnold Networks (KANs) with LMMs for the discovery and approximation of dynamical systems' vector fields. Specifically, we begin by establishing precise error bounds for two-layer B-spline KANs when approximating the governing functions of dynamical systems. Leveraging the approximation capabilities of KANs, we demonstrate that for certain families of LMMs, the total error is constrained within a specific range that accounts for both the method's step size and the network's approximation accuracy. Additionally, we analyze the difference between the numerical solution obtained from solving the ordinary differential equations with the fitted vector fields and the true solution of the dynamical system. To validate our theoretical results, we provide several numerical examples that highlight the effectiveness of our approach.

Keywords: Kolmogorov-Arnold network, linear multistep method, discovery of dynamics, error estimation, Vapnik-Chervonenkis (VC) dimension

1. Introduction

Dynamical systems are essential for understanding and predicting the evolution of physical phenomena. They encompass a spectrum of mathematical descriptions, from the simplicity of harmonic motion to the intricacies of fluid dynamics and turbulence, providing analytical tools crucial for assessing the historical dependence of system states and forecasting their future trajectories. Typically articulated through exact differential equations rooted in fundamental physical laws—such as conservation of energy, mass, and momentum-traditional first-principles approaches face limitations when applied to highly complex systems. In an era abundant with data and witnessing rapid strides in machine learning, a formidable challenge arises: the possibility of extracting detailed mathematical models directly from data for dynamical systems. This problem holds significant implications not only for theoretical physics and applied mathematics but also poses new challenges for engineering practice and the field of data science [1, 3, 10, 34].

^{*}Corresponding author. Email address: qguo@math.shnu.edu.cn

This problem can be summarized as follows: given certain discrete values of the state equation, how can we automatically discover complex dynamical systems from this data? Approaches to system identification generally fall into three major categories: model-driven methods, data-driven methods, and hybrid methods combining both. Data-driven methods focus more on capturing complex relationships, whereas model-driven methods prioritize the interpretability of the system structure. Hybrid methods aim to strike a balance between accuracy and physical consistency. In most real-world scenarios, methods typically involve a combination of these approaches. Representative methods include Gaussian processes, symbolic regression, sparse regression, statistical learning, and linear multistep methods (LMMs) using neural networks.

Gaussian processes [17, 24, 25] adeptly model system states or derivative output relationships using Gaussian distributions, optimizing hyperparameters by maximizing marginal log-likelihood to facilitate system parameter inference. While suitable for low to moderate-dimensional problems, they often require sparse approximations in high-dimensional settings and certain restrictions are imposed on the system form, typically used for parameter estimation in simpler systems [39]. Symbolic regression [4, 29] generates and refines symbolic models to fit observed data, providing physically meaningful functional forms for equation modeling [28]. However, for large or complex systems, it can be computationally expensive and prone to overfitting, often necessitating additional regularization techniques to ensure generalization capability. Sparse regression [6, 27, 38] approximates system control or state functions by identifying a sparse combination from a candidate basis function set, with coefficients determined by sparse regression algorithms. This approach offers an explicit system formula with minimal prior knowledge requirements [39], but its efficiency may be compromised or it may fail to accurately fit complex dynamical models if the system lacks a simple or sparse representation [20, 21, 28]. Statistical learning methods [21, 40] minimize empirical error to select an appropriate kernel function within a hypothesis space, learning the system's interaction relationships. This approach mitigates the curse of dimensionality to some extent and can be used to identify interaction patterns in high-dimensional systems [21], but it is limited to dynamical systems that can be approximately represented through kernel functions [11]. LMMs with neural networks [26, 35, 36] approximate control functions via neural networks, identifying control functions by minimizing the residuals of the discretized dynamical system through LMMs. This approach can achieve high convergence orders and handle complex or high-dimensional systems, as demonstrated in [20, 28, 39]. The stability and convergence of dynamic system identification have been thoroughly analyzed in [9, 15], where the relationship between the overall error, the error of the approximating network, and the residual of the LMM-discretized dynamic system was established.

In light of these considerations, this study delves into the convergence theory of the LMMs framework integrated with deep learning, employing Kolmogorov–Arnold Networks (KANs) as the neural network model. Through meticulous error analysis of KANs, we derive the discrepancy between the numerical solution $\boldsymbol{x}_{\mathcal{NN}}$ obtained from the learned network and the target state function \boldsymbol{x} .

The structure of the paper is outlined as follows: Sections 2.1 and 2.2 provide an introduction to the background of Kolmogorov–Arnold Networks (KANs) and the construction of the proposed network. Sections 2.3 and 2.4 detail the upper and lower error bounds for the proposed two-layer B-spline KANs, respectively. Section 3 offers an overview of dynamical systems and the foundational knowledge of Linear Multistep Methods (LMMs), followed by a description of the application of LMMs in dynamic identification. Section 4 presents a comprehensive error analysis of the LMMs when applied in conjunction with the two-layer B-spline KANs. Numerical experiments to validate the theoretical error results are conducted in Section 5.

2. B-Spline KANs Approximation

The theoretical foundation of traditional Multi-layer Perceptrons (MLPs) is often ascribed to the universal approximation theorem. This study, however, pivots attention towards the Kolmogorov-Arnold representation theorem, which serves as the theoretical basis for a neural network known as KANs. Through an in-depth analysis of the network's constituent basis functions, this research delineates a rigorous quantification of the approximation error within a two-layer KANs framework employing B-spline functions. The approximation is further grounded in Linear Multistep Methods (LMMs) for trajectory-based dynamical system identification, incorporating implicit regularization. Moreover, this study employs the concept of Vapnik-Chervonenkis (VC) dimension from computational learning theory to ascertain the minimum error bound for networks that leverage B-spline basis functions.

2.1. Kolmogorov-Arnold theorem

Braun and Griebel [5] have elucidated that any continuous multivariate function f defined on a bounded domain can be expressed as a finite sum of continuous univariate functions, with addition as the combining operation. In essence, this representation technique is applicable to continuous functions within the specified domain.

Lemma 2.1 (Kolmogorov-Arnold Representation theorem [5]). Let $f:[0,1]^d \to \mathbb{R}$ be an arbitrary multivariate continuous function. Then it has the representation

$$f(\mathbf{x}) = \sum_{q=0}^{2d} \phi_q \left(\sum_{p=1}^d \psi_{q,p}(x_p) \right) \quad \text{for any } \mathbf{x} = (x_1, \dots, x_d) \in [0, 1]^d$$
 (2.1)

with continuous one-dimensional outer and inner functions ϕ_q and $\psi_{q,p}$. All these functions ϕ_q , $\psi_{q,p}$ are defined on the real line. The inner functions $\psi_{q,p}$ are independent of the function f.

The foundational role of addition is emphasized by the demonstration that any multivariate function can be represented as a combination of univariate functions through addition. This finding may be seen as encouraging news for machine learning, suggesting that the challenge of learning high-dimensional functions could be simplified to learning a polynomial number of one-dimensional functions, potentially easing the complexity of the task. However, the situation is complicated by the fact that these one-dimensional functions may exhibit non-smooth or fractal characteristics, which pose significant challenges for their learnability in practice [18, 23], particularly for optimization algorithms. As a result, the Kolmogorov-Arnold representation theorem has largely been overlooked in the field of machine learning, viewed as theoretically sound yet often lacking practical applicability [18, 23].

However, we believe in the significant potential of the Kolmogorov-Arnold theorem for machine learning. First of all, we are not limited to the original formulation (2.1), which features a two-layer structure with nonlinearities and a limited number of terms, specifically (2d+1), in the hidden layer. We simplify the multi-layer KANs proposed in [19] by considering a two-layer KANs where the first layer has a width of dimension d and the second layer has a width of N. This allows us to obtain a precise estimation of the error for our network. The specific construction of the network will be detailed in the next subsection.

2.2. B-Spline KANs framework

Consider a scenario within the realm of supervised learning where the objective is to approximate a function f such that for a given set of input-output pairs $\{x_i, y_i\}$, the relationship $y_i \approx f(x_i)$ holds for every data point. Lemma 2.1 suggests that this objective can be achieved by identifying suitable univariate functions $\psi_{q,p}$ and ϕ_q . Kolmogorov-Arnold Networks (KANs), a class of neural networks predicated on the Kolmogorov-Arnold representation theorem, offer a novel approach to this end. According to the theorem, any continuous multivariate function $f(x) = f(x_1, x_2, ...)$ defined over a finite domain can be decomposed into a finite combination of continuous univariate functions, with addition serving as the binary operation that combines these elements.

The approximation for f(x) was proposed in [19] as follows:

$$f(\boldsymbol{x}) = \sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1, i_L, i_{L-1}} \left(\sum_{i_{L-2}=1}^{n_{L-2}} \cdots \left(\sum_{i_1=1}^{n_1} \phi_{1, i_1, i_0} \left(\sum_{i_0=1}^{n_0} \phi_{0, i_1, i_0}(x_{i_0}) \right) \right) \cdots \right). \tag{2.2}$$

The deep network in question is designed as a generalization of the Kolmogorov-Arnold representation theorem. Below, we will explain the formulas and concepts that appear in the equations, simplify them to obtain our two-layer B-spline KANs, and discuss the details of our networks implementation.

The main notations are listed as follows.

- Vectors and matrices are denoted in a bold font. Standard vectorization is adopted in the matrix and vector computation. For example, adding a scalar and a vector means adding the scalar to each entry of the vector.
- KAN layer with d_{in} -dimensional inputs and d_{out} -dimensional outputs can be defined as a matrix of 1D functions

$$\mathbf{\Phi} = \{\phi_{a,p}\}, \quad p = 1, 2, \dots, d_{in}, \quad q = 1, 2, \dots, d_{out}, \tag{2.3}$$

where the functions $\phi_{q,p}$ have trainable parameters, as detaild below. In the Kolmogov-Arnold theorem, the inner functions form a KAN layer with $d_{in}=d$ and $d_{out}=2d+1$, and the outer functions form a KAN layer with $d_{in}=2d+1$ and $d_{out}=1$. So the Kolmogorov-Arnold representations in (2.1) are simply compositions of two-layer KANs.

• The shape of a KAN is represented by an integer array

$$[d_0, d_1, \ldots, d_L],$$

where d_i is the number of nodes in the i^{th} layer of the computational graph. We denote the i^{th} neuron in the l^{th} layer by (l,i), and the activation value of the (l,i)-neuron by $x_{l,i}$. Between layer l and layer l+1, there are $n_l n_{l+1}$ activation functions: the activation function that connects (l,i) and (l+1,j) is denoted by

$$\phi_{l,i,i}, \quad l = 0, \dots, L - 1, \quad i = 1, \dots, n_l, \quad j = 1, \dots, n_{l+1}.$$
 (2.4)

• The pre-activation of $\phi_{l,j,i}$ is simply $x_{l,i}$, the post-activation of $\phi_{l,j,i}(x_{l,i})$ is denoted by $\tilde{x}_{l,j,i} \equiv \phi_{l,j,i}(x_{l,i})$. The activation value of the (l+1,j) neuron is simply the sum of all incoming postactivations:

$$x_{l+1,j} = \sum_{i=1}^{n_l} \tilde{x}_{l,j,i} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}), \quad j = 1, \dots, n_{l+1}.$$

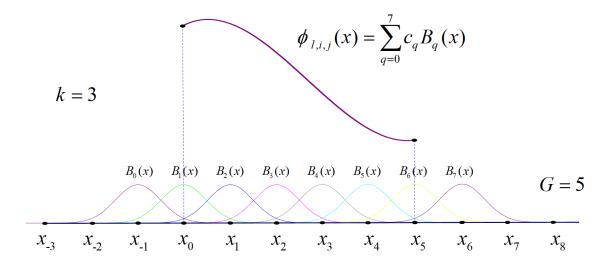


Figure 2.1. The activation function is parameterized by a set of B-spline basis functions.

In matrix form, this reads

$$\boldsymbol{x}_{l+1} = \underbrace{\begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix}}_{\boldsymbol{\Phi}_{l}} \boldsymbol{x}_{l}, \tag{2.5}$$

where Φ_l is the function matrix corresponding to the l^{th} KAN layer. A general KAN is a composition of L layers: given an input vector $\mathbf{x}_0 \in R^{n_0}$, the output of KAN is

$$KAN(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \dots \circ \Phi_1 \circ \Phi_0)x. \tag{2.6}$$

For the existing multilayer KAN, the main purpose of this paper is to study the lower bound of the approximation error of any continuous function f over a given finite interval using B-spline functions with a specified number of layers. This is related to the number of grid points G of the constructed B-spline basis functions, as well as the degree of the B-splines. Below, we will first introduce the structure of the two-layer B-spline KAN we are considering, and in the next section, we will provide an exact expression for our error lower bound and prove it.

2.3. B-Spline KANs approximation

Let's review the two-layer B-spline KANs earlier. Mathematically speaking,

$$\phi(\mathbf{x}) = \sum_{j=1}^{N} \phi_{1,j} \left(\sum_{i=1}^{d} \phi_{0,j,i}(x_i) \right) \quad \text{for any } \mathbf{x} = (x_1, \dots, x_d) \in [0, 1]^d,$$
 (2.7)

where $\phi_{0,j,i}$ represents the inner B-spline activation function, collectively denoted as $\Phi_0(\mathbf{x})$. $\phi_{1,j}$ denotes the outer B-spline activation function, also collectively denoted as $\Phi_1(\mathbf{x})$, where each B-spline activation function is similarly represented by a set of basis functions, $\phi_{l,i,j}(\mathbf{x}) = \sum_{q=0}^{G+k-1} c_q B_q(\mathbf{x})$.

 $x_{l,i}$ represents the *i*-th element in the *l*-th layer, where l=1,2 and $i=1,2,\ldots,N$. Next, we will derive the approximation properties of the KANs for the above network structure. For convenience, we present all notations used throughout this dissertation.

Definition 2.1 (Modulus of continuity). Given $S \subseteq \mathbb{R}^d$, we can define the modulus of continuity of a continuous function f by

$$\omega_f^S(r) = \sup\{|f(x) - f(y)| : ||x - y||_2 \le r, x, y \in S\} \text{ for any } r \ge 0.$$
 (2.8)

Clearly, $\omega_f^S(cr) \leq c\omega_f^S(r)$ for any $c \in \mathbb{N}^+$ and $r \geq 0$.

Definition 2.2 (Hölder continuous functions). Given $E \subseteq \mathbb{R}^d$, a function $f: E \to \mathbb{R}$ is called Hölder continuous if there exist constants $\lambda > 0$ and $\alpha \in (0,1]$ such that for all $x, y \in E$,

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le \lambda ||\boldsymbol{x} - \boldsymbol{y}||^{\alpha},$$

which can be denoted as the class $H_{\lambda}(C^{\alpha}(E))$. Here, $\|\cdot\|$ represents the distance between \boldsymbol{x} and \boldsymbol{y} (typically the Euclidean distance). In particular, $\alpha=1$, the Hölder continuous function is referred to as a Lipschitz continuous function, meaning there exists a constant L such that:

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \le L \|\boldsymbol{x} - \boldsymbol{y}\|$$
 for all $\boldsymbol{x}, \boldsymbol{y} \in E$.

Lemma 2.2 (Extension of Continuous Functions [30]). Suppose f is a uniformly continuous function defined on a subset $E \subset S$, where S is a metric space with a metric $d_S(\cdot, \cdot)$, then there exists a uniformly continuous function g on S such that $f(\mathbf{x}) = g(\mathbf{x})$ for $\mathbf{x} \in E$ and $\omega_f^E(r) = \omega_g^S(r)$ for any $r \geq 0$.

Lemma 2.3 (B-spline function error approximation [8]). Let f(x) be a continuous functions on [a, b], there exists a function $\phi(x) = \sum_{q=0}^{G+k-1} c_q B_q(x)$ implemented by B-spline basis functions $B_{i,k}(x)$ with degree k of the piecewise B-spline polynomial and number of knots G such that

$$||f(x) - \phi(x)|| \le \omega_f \left(\min \left\{ \frac{b - a}{\sqrt{2k - 2}}, \sqrt{\frac{k}{12}} \cdot \frac{b - a}{G} \right\} \right), \tag{2.9}$$

where $B_{i,k}(x)$ can be uniquely defined using a recurrence relation. Frist, we have

$$B_{i,0}(x) = \begin{cases} 1, & x_i \le x < x_{i+1}, \\ 0, & otherwise. \end{cases}$$

Then we have

$$B_{i,k}(x) = \frac{x - x_i}{x_{i+k} - x_i} B_{i,k-1}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} B_{i+1,k-1}(x)$$

and the basis functions constructed in this way are uniquely defined.

Theorem 2.1 (Upper bound of two-layer B-spline KANs). Let f be a continuous functions on $[0,1]^d$, suppose that a function $f(\mathbf{x})$ admits a representation

$$f(\boldsymbol{x}) = (\boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0) \boldsymbol{x},$$

for any $k, G \in \mathbb{N}^+$, there exists a function $\phi(\mathbf{x}) = (\mathbf{\Phi}_1^G \circ \mathbf{\Phi}_0^G)\mathbf{x}$, implemented by a two-layer B-spline KANs with degree k of the piecewise B-spline polynomial and number of knots G such that

$$||f(\boldsymbol{x}) - \phi(\boldsymbol{x})|| \le N(L_G d + 1) \cdot \omega_f \left(\min \left\{ \frac{1}{\sqrt{2k - 2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right), \tag{2.10}$$

where d denotes the dimension of the input vector, N represents the number of basis functions in the second-layer network, and L_G is the Lipschitz constant of $\phi^G(\mathbf{x})$.

Proof. To establish the inequality in equation (2.11), we start by decomposing it into two parts:

$$\left\| (\boldsymbol{\Phi}_{1} \circ \boldsymbol{\Phi}_{0}) \boldsymbol{x} - (\boldsymbol{\Phi}_{1}^{G} \circ \boldsymbol{\Phi}_{0}^{G}) \boldsymbol{x} \right\| \leq \left\| (\boldsymbol{\Phi}_{1} \circ \boldsymbol{\Phi}_{0}) \boldsymbol{x} - (\boldsymbol{\Phi}_{1}^{G} \circ \boldsymbol{\Phi}_{0}) \boldsymbol{x} \right\| + \left\| (\boldsymbol{\Phi}_{1}^{G} \circ \boldsymbol{\Phi}_{0}) \boldsymbol{x} - (\boldsymbol{\Phi}_{1}^{G} \circ \boldsymbol{\Phi}_{0}^{G}) \boldsymbol{x} \right\|.$$

$$(2.11)$$

We first address the first term. Let $\Phi_0 x = y$. By invoking the classical 1D B-spline theory from Lemma 2.3, we obtain:

$$\begin{split} \left\| (\boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0) \boldsymbol{x} - (\boldsymbol{\Phi}_1^G \circ \boldsymbol{\Phi}_0) \boldsymbol{x} \right\| &= \left\| \sum_{j=1}^N \phi_{1,j}(\boldsymbol{y}) - \sum_{j=1}^N \phi_{1,j}^G(\boldsymbol{y}) \right\| \\ &\leq N \cdot \omega_f \left(\min \left\{ \frac{1}{\sqrt{2k-2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right). \end{split}$$

Next, we consider the second term on the right-hand side of (2.11). Utilizing the Lipschitz constant L_G of $\phi^G(x)$ and $\Phi_0^G x = \bar{y}$, we have:

$$\begin{split} \left\| (\boldsymbol{\Phi}_{1}^{G} \circ \boldsymbol{\Phi}_{0}) \boldsymbol{x} - (\boldsymbol{\Phi}_{1}^{G} \circ \boldsymbol{\Phi}_{0}^{G}) \boldsymbol{x} \right\| &\leq \left\| \sum_{j=1}^{N} \phi_{1,j}^{G}(\boldsymbol{y}) - \sum_{j=1}^{N} \phi_{1,j}^{G}(\bar{\boldsymbol{y}}) \right\| \\ &\leq \left\| \sum_{j=1}^{N} (\phi_{1,j}^{G}(\boldsymbol{y}) - \phi_{1,j}^{G}(\bar{\boldsymbol{y}})) \right\| \\ &\leq NL_{G} \|\boldsymbol{y} - \bar{\boldsymbol{y}}\| = NL_{G} \left\| \boldsymbol{\Phi}_{0} \boldsymbol{x} - \boldsymbol{\Phi}_{0}^{G} \boldsymbol{x} \right\| \\ &\leq NL_{G} d \cdot \omega_{f} \left(\min \left\{ \frac{1}{\sqrt{2k-2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right). \end{split}$$

Given the definitions:

$$\Phi_{0,j}(\boldsymbol{x}) = \sum_{i=1}^d \phi_{0,j,i}(x_i), \quad \Phi_{0,j}^G(\boldsymbol{x}) = \sum_{i=1}^d \phi_{0,j,i}^G(x_i),$$

we can express:

$$\begin{split} \boldsymbol{\Phi}_0 \boldsymbol{x} &= [\Phi_{0,1}(\boldsymbol{x}), \Phi_{0,2}(\boldsymbol{x}), \dots, \Phi_{0,d}(\boldsymbol{x})]^{\mathrm{T}}, \\ \boldsymbol{\Phi}_0^G \boldsymbol{x} &= [\Phi_{0,1}^G(\boldsymbol{x}), \Phi_{0,2}^G(\boldsymbol{x}), \dots, \Phi_{0,d}^G(\boldsymbol{x})]^{\mathrm{T}}, \\ \boldsymbol{\Phi}_0 \boldsymbol{x} - \boldsymbol{\Phi}_0^G \boldsymbol{x} &= [\Phi_{0,1}(\boldsymbol{x}) - \Phi_{0,1}^G(\boldsymbol{x}), \Phi_{0,2}(\boldsymbol{x}) - \Phi_{0,2}^G(\boldsymbol{x}), \dots, \Phi_{0,d}(\boldsymbol{x}) - \Phi_{0,d}^G(\boldsymbol{x})]^{\mathrm{T}}, \\ \left\| \boldsymbol{\Phi}_0 \boldsymbol{x} - \boldsymbol{\Phi}_0^G \boldsymbol{x} \right\| \leq d \cdot \omega_f \left(\min \left\{ \frac{1}{\sqrt{2k-2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right). \end{split}$$

Returning to (2.11), we combine the results:

$$\begin{split} \left\| (\boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0) \boldsymbol{x} - (\boldsymbol{\Phi}_1^G \circ \boldsymbol{\Phi}_0^G) \boldsymbol{x} \right\| &\leq N \cdot \omega_f \left(\min \left\{ \frac{1}{\sqrt{2k-2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right) \\ &+ N L_G d \cdot \omega_f \left(\min \left\{ \frac{1}{\sqrt{2k-2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right) \\ &\leq N (L_G d + 1) \cdot \omega_f \left(\min \left\{ \frac{1}{\sqrt{2k-2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right), \end{split}$$

which satisfies equation (2.10).

Corollary 2.1. Let f be a continuous functions on $[-R, R]^d$, where R is an arbitrary positive real number, suppose that a function f(x) admits a representation

$$f(\boldsymbol{x}) = (\boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0) \boldsymbol{x},$$

for any $k, G \in \mathbb{N}^+$, there exists a function $\phi(\mathbf{x}) = (\mathbf{\Phi}_1^G \circ \mathbf{\Phi}_0^G)\mathbf{x}$, implemented by a two-layer B-spline KANs with degree k of the piecewise B-spline polynomial and number of knots G such that

$$||f(\boldsymbol{x}) - \phi(\boldsymbol{x})|| \le N(L_G d + 1) \cdot \omega_f \left(\min \left\{ \frac{2R}{\sqrt{2k - 2}}, \sqrt{\frac{k}{12}} \cdot \frac{2R}{G} \right\} \right), \tag{2.12}$$

where d denotes the dimension of the input vector, N represents the number of basis functions in the second-layer network, and L_G is the Lipschitz constant of $\phi^G(\mathbf{x})$.

Proof. Let $f \in C(E)$ be an arbitrary component of f and f is a continuous function. By Lemma 2.2 setting $S = \mathbb{R}^d$, there exists $h \in C(\mathbb{R}^d)$ such that h(x) = f(x) for any $x \in E \subseteq [-R, R]^d$ and $\omega_h^S(r) = \omega_f^E(r)$ for any $r \geq 0$. Define

$$\hat{h}(\boldsymbol{x}) := h(2R\boldsymbol{x} - R)$$
 for any $\boldsymbol{x} \in \mathbb{R}^d$.

According to Theorem 2.1 there exist $\hat{\phi}(x)$ such that

$$\left\|\hat{h}(\boldsymbol{x}) - \hat{\phi}(\boldsymbol{x})\right\| \leq N(Ld+1) \cdot \omega_{\hat{h}}^S \left(\min\left\{\frac{1}{\sqrt{2k-2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G}\right\}\right), \quad \text{for any } \boldsymbol{x} \in [0,1]^d.$$

By implementing a variable transformation to revert our notation from \hat{h} to h, we establish the relationship $f(\mathbf{x}) = h(\mathbf{x}) = \hat{h}\left(\frac{\mathbf{x}+R}{2R}\right)$ for any $\mathbf{x} \in E \subseteq [-R,R]^d$. Accordingly, the modulus of \hat{h} , denoted as $\omega_{\hat{h}}^S(r)$, is equivalent to $\omega_{\hat{h}}^S(2Rr)$, which in turn equals $\omega_f^E(2Rr)$.

For the function $\phi(\boldsymbol{x})$, we define it in terms of $\hat{\phi}$ as $\phi(\boldsymbol{x}) := \hat{\phi}\left(\frac{\boldsymbol{x}+R}{2R}\right)$ for $\frac{\boldsymbol{x}+R}{2R} \in [0,1]^d$. This leads us to the following inequality:

$$||f(\boldsymbol{x}) - \phi(\boldsymbol{x})|| = ||h(\boldsymbol{x}) - \phi(\boldsymbol{x})|| = \left\| \hat{h} \left(\frac{\boldsymbol{x} + R}{2R} \right) - \hat{\phi} \left(\frac{\boldsymbol{x} + R}{2R} \right) \right\|$$

$$\leq N(L_G d + 1) \cdot \omega_{\hat{h}}^S \left(\min \left\{ \frac{2R}{\sqrt{2k - 2}}, \sqrt{\frac{k}{12}} \cdot \frac{2R}{G} \right\} \right)$$

$$\leq N(L_G d + 1) \cdot \omega_f \left(\min \left\{ \frac{2R}{\sqrt{2k - 2}}, \sqrt{\frac{k}{12}} \cdot \frac{2R}{G} \right\} \right),$$

which implies (2.12).

Theorem 2.2. Let f be a function that is Hölder continuous with exponent α and constant λ on the domain $[0,1]^d$. Assume that $f(\mathbf{x})$ can be expressed as a composition of two functions:

$$f(\boldsymbol{x}) = (\boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0) \boldsymbol{x}.$$

For any positive integers k and G, there exists an approximating function $\phi(\mathbf{x}) = (\mathbf{\Phi}_1^G \circ \mathbf{\Phi}_0^G)\mathbf{x}$, realizable by a two-layer B-spline KAN with degree k and G knots, such that the approximation error is bounded by:

$$||f(\boldsymbol{x}) - \phi(\boldsymbol{x})|| \le \lambda N(L_G d + 1) \left(\min \left\{ \frac{1}{\sqrt{2}} (k - 1)^{-\frac{1}{2}}, \sqrt{\frac{1}{12}} G^{-1} k^{\frac{1}{2}} \right\} \right)^{\alpha},$$
 (2.13)

where d is the dimensionality of the input vector, N is the count of basis functions in the second layer of the network, and L_G is the Lipschitz constant of $\phi^G(\mathbf{x})$.

Proof. Hölder continuous functions of order α with a Hölder continuous constant λ on $[0,1]^d$ can be denoted as the class $H_{\lambda}\left(C^{\alpha}\left([0,1]^d\right)$, so we have $\omega_f(r) \leq \lambda r^{\alpha}$. According to Theorem 2.1

$$||f(\boldsymbol{x}) - \phi(\boldsymbol{x})|| \leq N(L_G d + 1) \cdot \omega_f \left(\min \left\{ \frac{1}{\sqrt{2k - 2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right)$$

$$\leq \lambda N(L_G d + 1) \cdot \left(\min \left\{ \frac{1}{\sqrt{2k - 2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right)^{\alpha}$$

$$\leq \lambda N(L_G d + 1) \left(\min \left\{ \frac{1}{\sqrt{2}} (k - 1)^{-\frac{1}{2}}, \sqrt{\frac{1}{12}} G^{-1} k^{\frac{1}{2}} \right\} \right)^{\alpha},$$

thus, equation (2.13) holds.

Corollary 2.2. Let f be a Hölder continuous functions of order α with a Hölder continuous constant λ on $[-R, R]^d$, Suppose that a function $f(\mathbf{x})$ admits a representation

$$f(\boldsymbol{x}) = (\boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0) \boldsymbol{x}$$

for any $k, G \in \mathbb{N}^+$, there exists a function $\phi(\mathbf{x}) = (\mathbf{\Phi}_1^G \circ \mathbf{\Phi}_0^G)\mathbf{x}$, implemented by a two-layer B-spline KANs with degree k of the piecewise B-spline polynomial and number of knots G such that

$$||f(\boldsymbol{x}) - \phi(\boldsymbol{x})|| \le \lambda N(L_G d + 1)R^{\alpha} \left(\min \left\{ \sqrt{2}(k-1)^{-\frac{1}{2}}, \sqrt{\frac{1}{3}}G^{-1}k^{\frac{1}{2}} \right\} \right)^{\alpha},$$
 (2.14)

where d denotes the dimension of the input vector, N represents the number of basis functions in the second-layer network, and L_G is the Lipschitz constant of $\phi^G(\mathbf{x})$.

2.4. Lower bound approximation error and VC-dimension

The approximation error and the Vapnik-Chervonenkis (VC) dimension are pivotal metrics for evaluating the capacity, or complexity, of a function class. This subsection delves into the interplay between these two fundamental concepts [14].

We commence by elucidating the definitions of VC-dimension and associated terminologies.

Definition 2.3 (Shattering [2]). Let S represent a class of functions mapping from a domain \mathbb{E} to $\{0,1\}$. The class S is said to shatter a subset of points $\{x_1, x_2, \dots, x_m\} \subseteq \mathbb{E}$, if

$$\left| \left\{ \left[s(x_1), s(x_2), \cdots, s(x_m) \right]^T \in \{0, 1\}^m : s \in S \right\} \right| = 2^m,$$

where $|\cdot|$ denotes the size of the set.

The above definition means, given any $\xi_i \in \{0,1\}$ for $i = 1, 2, \dots, m$, there exists $s \in S$ such that $s(\boldsymbol{x_i}) = \xi_i$ for all i. For a general function set \mathcal{A} with its elements mapping from \mathbb{E} to \Re , we say \mathcal{A} shatters $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_m}\} \subseteq \mathbb{E}$, if $\mathcal{D} \circ \mathcal{A}$ does, where

$$\mathcal{D}(t) := \begin{cases} 1, & t \ge 0, \\ 0, & t < 0 \end{cases} \text{ and } \mathcal{D} \circ \mathcal{A} := \{ \mathcal{D} \circ f : f \in \mathcal{A} \}.$$

Definition 2.4 (Growth function [2]). Assume \mathcal{A} is a class of functions mapping from a general domain \mathbb{E} to $\{0,1\}$, the growth function of \mathcal{A} is defined as

$$\Pi_{\mathcal{A}}(m) := \max_{\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_m}} \left| \left\{ \left[h(\boldsymbol{x_1}), h(\boldsymbol{x_2}), \cdots, h(\boldsymbol{x_m}) \right]^T \in \left\{ 0, 1 \right\}^m : h \in \mathcal{A} \right\} \right|.$$

Definition 2.5 (VC-dimension [2]). Assume \mathcal{A} is a class of functions from \mathbb{E} to $\{0,1\}$. The VC-dimension of \mathcal{A} , denoted by VCDim(\mathcal{A}), is the size of the largest shattered set, namely,

$$VCDim(A) := \sup \{ m \in \mathbb{N}^+ : \Pi_A(m) = 2^m \}.$$

If there is no largest m, $VCDim(A) = \infty$.

Let \mathcal{A} be a class of functions from \mathbb{E} to \Re . The VC-dimension of \mathcal{A} , denoted by VCDim(\mathcal{A}), is defined by VCDim(\mathcal{A}) := VCDim($\mathcal{D} \circ \mathcal{A}$) [2]. In particular, the expression "VC-dimension of a network (architecture)" means the VC-dimension of the function set that consists of all functions implemented by this network architecture.

Lemma 2.4 (Theorem 2.4 of [33]). Assume A is a function set with all elements defined on $[0,1]^d$. Given any $\varepsilon > 0$, suppose VCDim(A) ≥ 1 and

$$\inf_{\phi \in \mathcal{A}} \|\phi - f\|_{L^{\infty}([0,1]^d)} \le \varepsilon, \quad \text{for any } f \in H_1(C^{\alpha}([0,1]^d)).$$

Then $VCDim(A) \ge (9\varepsilon)^{-d/\alpha}$.

As shown in [22, 30, 31], VC-dimension essentially determines the lower bound of the approximation errors of networks. Next, we introduce the relationship between two approximation errors and the VC-dimension.

Lemma 2.4 investigates the connection between VC-dimension of \mathcal{A} and the approximation errors of functions in $H_1(C^{\alpha}([0,1]^d))$ approximated by elements of \mathcal{A} . Denote the best approximation error of functions in $H_1(C^{\alpha}([0,1]^d))$ approximated by the elements of \mathcal{A} as

$$\varepsilon_{\alpha,d}(\mathcal{A}) := \sup_{f \in H_1(C^{\alpha}([0,1]^d))} \left(\inf_{\phi \in \mathcal{A}} \|\phi - f\|_{L^{\infty}([0,1]^d)} \right).$$

Subsequently, by using Lemma 2.4, we establish the inequality

$$VCDim(\mathcal{A})^{-\alpha/d}/9 \le \varepsilon_{\alpha,d},$$

which implies that a substantial VC-dimension is essential for achieving a favorable approximation rate. This is because the optimal approximation rate is governed by a factor that is explicitly dependent on the VC-dimension. Moving forward, we shall examine the VC-dimension of the network we have constructed.

Lemma 2.5 (Theorem 8 of [2]). Consider a neural network with M parameters and U units with activation functions that are piecewise polynomials with at most P pieces and of degree at most d. Let A be the set of (real-valued) functions computed by this network. Then $VCDim((\mathcal{D} \circ A)) = O(MU \log((d+1)P))$.

Theorem 2.3 (Lower bound of two-layer B-spline KANs). Let \mathbf{f} be a continuous functions on $[0,1]^d$, suppose that a function \mathbf{f} admits a representation

$$f = (\mathbf{\Phi}_1 \circ \mathbf{\Phi}_0) \mathbf{x}$$

for any $k, G \in \mathbb{N}^+$, there exists a function $\phi(\mathbf{x}) = (\mathbf{\Phi}_1^G \circ \mathbf{\Phi}_0^G)\mathbf{x}$, implemented by a two-layer B-spline KANs with degree k of the piecewise B-spline polynomial and number of knots G, P = G + k - 1 is the number of B-spline basis functions used for each basis function. The best approximation error for functions in $H_1(C^{\alpha}([0,1]^d))$, when approximated by elements of A, has a lower bound such that

$$\varepsilon_{\alpha,d}(\mathcal{A}) \ge C \Big(NP(d+1)(d+N+1) \log((d+1)P) \Big)^{-\alpha/d}, \tag{2.15}$$

where d denotes the dimension of the input vector, N represents the number of basis functions in second-layer B-spline KANs.

Proof. According to Lemma 2.5 a neural network with M parameters and U units with activation functions that are piecewise polynomials with at most P pieces and of degree at most d. Let \mathcal{A} be the set of functions computed by this network. Then

$$VCDim(\mathcal{A}) = VCDim(\mathcal{D} \circ \mathcal{A}) = O(MU \log ((d+1)P)) = CMU \log ((d+1)P)),$$

for our two-layer B-spline KANs, the total number of parameters is M = (dN + N)P, and the number of units is U = d + N + 1. Thus, the error in the theorem can be expressed in the following form

$$VCDim(\mathcal{A}) = VCDim(\mathcal{D} \circ \mathcal{A}) = CP(dN + N)(d + N + 1)\log((d + 1)P))$$

$$\leq CNP(d + 1)(d + N + 1)\log((d + 1)P)),$$

so we have

$$\varepsilon_{\alpha,d}(\mathcal{A}) \ge \text{VCDim}(\mathcal{A})^{-\alpha/d}/9$$

$$\ge C \Big(NP(d+1)(d+N+1) \log((d+1)P) \Big)^{-\alpha/d},$$

hence, we have (2.15).

A high VC-dimension is a prerequisite for achieving a favorable approximation error; however, it is not a sufficient condition on its own. The approximation error is also contingent upon additional structural attributes of the hypothesis space \mathcal{A} , for instance,

VCDim
$$(\{\phi: \phi(x) = \cos(ax), a \in R\}) = \infty.$$

However, this set fails to achieve a satisfactory approximation error when approximating Hölder continuous functions, as demonstrated in [32]. Therefore, constructing a hypothesis space with a large VC-dimension is a fundamental step towards achieving an effective approximation tool, but realizing the full potential of approximation capabilities necessitates a more nuanced design of the hypothesis space. This refined design philosophy has been central to our work in this paper, which posits that a well-crafted hypothesis space is a necessary condition for our Kolmogorov–Arnold Networks (KANs) to achieve superior approximation capabilities.

3. Dynamical systems and discovery of dynamics

In this section, we first present the essential notations and definitions, consistent with the conventions established in [15]. For a more exploration of LMMs, refer to [7]. Next, we introduce a numerical scheme for an inverse problem in the discovery of dynamical systems using LMMs. Mathematically, this involves solving for the values of \boldsymbol{f} given \boldsymbol{x} , resulting in a system of linear equations. We then analyze the numerical error associated with this system.

3.1. LMMs: Notation and concepts

Suppose d > 0 is the dimension of the dynamics, consider the ordinary differential equation (ODE)

$$\frac{\mathrm{d}}{\mathrm{d}t} \mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)), \quad 0 \le t \le T,
\mathbf{x}(0) = \mathbf{x}_0.$$
(3.1)

Let $\boldsymbol{x} \in C^{\infty}[0,T]^d$ be an unknown vector-valued state function, $\boldsymbol{f}: \mathbb{R}^d \to \mathbb{R}^d$ be a given vector-valued governing function, $\boldsymbol{x}_0 \in \mathbb{R}^d$ be a given initial vector. To seek a numerical solution, we assume a grid on the interval [0,T] defined to be a set of points: $0=t_0 < t_1 < \cdots < t_{N_1} = T$ with equidistant mesh $t_{n+1}-t_n=h=\frac{T}{N_1}, n\in\{0,1,\ldots,N_1\}$. Let $[0,T]_h$ denote this ordered set. We denote the set of grid functions $\Gamma_h[0,T]=\left\{\boldsymbol{x}|\boldsymbol{x}\in\mathbb{R}^{(N_1+1)\times d}, \boldsymbol{x}_n=\boldsymbol{x}(t_n)\in\mathbb{R}^d, t_n\in[0,T]_h\right\}$, the objective for solving the initial value problem is to find an approximate value $\boldsymbol{x}_n\approx\boldsymbol{x}(t_n)$ for each n when $\boldsymbol{f}(\boldsymbol{x})$ is given [7].

An M-step LMM approximates the n-th value $\mathbf{x}_n = \mathbf{x}(t_n)$ in terms of the previous $M(M \ge 1)$ time steps $\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \dots, \mathbf{x}_{n-M}$ [7, 37]. An M-step linear multistep method is given by

$$\sum_{m=0}^{M} \alpha_m \mathbf{x}_{n-m} = h \sum_{m=0}^{M} \beta_m f(\mathbf{x}_{n-m}), \quad n = M, M+1, \dots, N_1,$$
(3.2)

where $\boldsymbol{x} \in \Gamma_h[0,T]$, the coefficients $\alpha_m, \beta_m \in \mathbb{R}$ for $m=0,1,\ldots,M$, and and α_0 is always nonzero. By the scheme, all \boldsymbol{x}_n are evaluated iteratively from n=M to $n=N_1$. In each step, $\boldsymbol{x}_{n-M},\ldots,\boldsymbol{x}_{n-1}$ are all given or computed previously such that \boldsymbol{x}_n can be computed by solving algebraic equations. If $\beta_0=0$, the scheme is called explicit since \boldsymbol{x}_n does not appear on the right-hand side and \boldsymbol{x}_n can be computed directly by

$$\boldsymbol{x}_n = \alpha_0^{-1} \sum_{m=1}^{M} \left(h \beta_m f(\boldsymbol{x}_{n-m}) - \alpha_m \boldsymbol{x}_{n-m} \right).$$

Otherwise, the scheme is called implicit and it requires solving nonlinear equations for x_n . The first value x_0 is simply set as $x_0 = x(t_0)$, while other initial values x_1, \ldots, x_{M-1} need to be computed by other approaches before performing the LMMs if M > 1. Common types of LMMs include Adams-Bashforth (AB) schemes, Adams-Moulton (AM) schemes, and backwards differentiation formula (BDF) schemes.

Next, we introduce the definition of the convergence order of numerical methods, along with the corresponding definitions involved.

Definition 3.1 (Residual operator [7]). Let $\hat{x} \in \Gamma_h[0,T]$. We define the numerical residual operator associated to (3.2) as

$$(R_h \hat{\boldsymbol{x}})_n := \frac{1}{h} \sum_{m=0}^{M} \alpha_m \hat{\boldsymbol{x}}_{n-m} - \sum_{m=0}^{M} \beta_m \boldsymbol{f}(\hat{\boldsymbol{x}}_{n-m}), \quad n = M, M+1, \dots, N_1.$$
 (3.3)

Definition 3.2 (Local truncation error [7]). Let $x \in \Gamma_h[0,T]$ be the exact solution of the dynamic system (3.1) defined at the grid coordinates. The local truncation error

$$T(t, \mathbf{x}; h)_n = \frac{1}{h} \sum_{m=0}^{M} \alpha_m \mathbf{x}_{n-m} - \sum_{m=0}^{M} \beta_m \mathbf{f}(\mathbf{x}_{n-m}), \quad n = M, M+1, \dots, N_1.$$
 (3.4)

Definition 3.3 (Order of error [7]). A linear multistep method (3.2) has error order p if, for a chosen vector norm $\|\cdot\|$, there exists a real constant C > 0 such that

$$\|\boldsymbol{T}(t,\boldsymbol{x};h)\| \le Ch^p$$

where $\boldsymbol{T}(t, \boldsymbol{x}; h) = \left(T(t, \boldsymbol{x}; h)_M, T(t, \boldsymbol{x}; h)_{M+1}, \dots, T(t, \boldsymbol{x}; h)_{N_1}\right) \in \mathbb{R}^{(N_1 - M + 1) \times d}$ and C is independent on t, \boldsymbol{x}, h .

3.2. Discovery of dynamics

The discovery of dynamics can be viewed as an inverse process of solving a dynamical system defined by (3.1). Given information on the state x at equidistant time steps $\{t_n\}_{n=0}^{N_1}$, our goal is to recover f, the governing function of the state.

Let $\mathbf{x}(t) \in C^{\infty}([0,T])^d$ and $\mathbf{f}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ be two vector-valued functions that are both unknown. Given the observations $\mathbf{x}_n = \mathbf{x}(t_n)$ for $n = 0, \dots, N$, we aim to determine $\mathbf{f}(\cdot)$. An effective method is to establish a discrete relationship between \mathbf{x}_n and $\mathbf{f}_n \approx \mathbf{f}(\mathbf{x}_n)$ using LMMs [15], expressed as:

$$h \sum_{m=0}^{M} \beta_m \mathbf{f}_{n-m} = \sum_{m=0}^{M} \alpha_m \mathbf{x}_{n-m}, \quad n = M, M+1, \dots, N_1.$$
 (3.5)

Here, $f_n \in \mathbb{R}^d$ serves as an approximation of $f(x_n)$. This discovery process is essentially the inverse of solving the dynamical system. To simplify, we focus on a scalar system, resulting in the following equation:

$$h\sum_{m=0}^{M} \beta_m f_{n-m} = \sum_{m=0}^{M} \alpha_m x_{n-m}, \quad n = M, M+1, \dots, N_1.$$
(3.6)

It is important to note that f_n may not be present in (3.6) for some indices n between 0 and N_1 . For instance, in AB schemes, f_{N_1} is not included in (3.6) because $\beta_0 = 0$. Generally, for a given LMM, we define r and q as the first and last indices where f_r and f_q appear in (3.6) with non-zero coefficients (i.e., β_{M-r} and β_{N_1-q} are both non-zero). We denote $\tau := q - r + 1$ as the total number of f_n involved in (3.6). For each linear M-step method, it is supposed to compute all unknowns $\{f_n\}_{n=r}^q$ by the linear relation (3.6). We write

$$\overrightarrow{\boldsymbol{f}_h} := \begin{bmatrix} f_r \ f_{r+1} \ \cdots \ f_q \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{\tau},$$

$$\overrightarrow{\boldsymbol{b}_h} := \frac{1}{h} \begin{bmatrix} \sum_{m=0}^{M} \alpha_m x_{M-m} & \sum_{m=0}^{M} \alpha_m x_{M+1-m} & \cdots & \sum_{m=0}^{M} \alpha_m x_{N_1-m} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{N_1-M+1},$$

$$\boldsymbol{B}_h := \begin{bmatrix} \beta_{M-r} \ \beta_{M-r-1} & \cdots & \beta_{N_1-q} \\ \beta_{M-r} & \beta_{M-r-1} & \cdots & \beta_{N_1-q} \\ & \ddots & \ddots & \ddots \\ & & & \beta_{M-r} & \beta_{M-r-1} & \cdots & \beta_{N_1-q} \end{bmatrix} \in \mathbb{R}^{(N_1-M+1)\times \tau}.$$

Then (3.5) leads to the following linear system

$$B_h \overrightarrow{f_h} = \overrightarrow{b_h}.$$
 (3.7)

In (3.5), the number of equations may not match the number of unknowns. For AB and AM schemes, it is insufficient to determine $\{f_n\}_{n=r}^q$ because there are fewer equations than unknowns. This results in the linear system (3.7) being underdetermined.

To address this issue, we can introduce $\Omega_a := q - (N_1 - M + 1)$ auxiliary linear conditions to ensure $\{f_n\}_{n=r}^q$ is uniquely determined. For instance, we can compute Ω_a unknown f_n values directly using a first-order (derivative) finite difference method (FDM) based on related data. To maintain consistency, the chosen FDM should have the same error order as the LMM. Assuming the LMM has order p, a straightforward approach is to calculate the initial Ω_a unknowns using a FDM of order p,

$$f_n = \frac{1}{h} \sum_{m=0}^{p} \mu_m x_{n+m}, \quad n = r, r+1, \dots, r+\Omega_a - 1,$$
 (3.8)

where μ_m are the corresponding finite difference coefficients. Note that (3.8) has the error estimate

$$\max_{r \le n \le r + \Omega_a - 1} |f_n - f(\boldsymbol{x}(t_n))| = O(h^p), \quad \text{as } h \to 0.$$

Then combining (3.5) and (3.8) leads to the augmented linear system

$$\mathbf{A}_{h}\overrightarrow{\mathbf{f}_{h}} = \begin{bmatrix} \mathbf{c}_{h} \\ \overrightarrow{\mathbf{b}_{h}} \end{bmatrix}, \tag{3.9}$$

where

$$egin{aligned} oldsymbol{c}_h &:= rac{1}{h} \left[\sum_{m=0}^p \mu_m x_{s+m} & \sum_{m=0}^p \mu_m x_{s+1+m} & \cdots & \sum_{m=0}^p \mu_m x_{s+\Omega_a-1+m}
ight]^T \in \mathbb{R}^{\Omega_a}, \ oldsymbol{A}_h &:= \left[egin{array}{c} oldsymbol{C} & oldsymbol{B}_h \end{array}
ight] & ext{and} & oldsymbol{C} &:= \left[egin{array}{c} oldsymbol{I}_{\Omega_a} & oldsymbol{O} \end{array}
ight] \end{aligned}$$

with I_{Ω_a} being the $\Omega_a \times \Omega_a$ identity matrix and O being the zero matrix of size $\Omega_a \times (\tau - \Omega_a)$. Clearly, (3.9) has a unique solution since the coefficient matrix is lower triangular with nonzero diagonals. Moreover, if $M \ll N_1$, the linear system (3.9) is sparse.

In general, as noted in [15], we can define auxiliary conditions in various ways beyond what has been discussed. Different types of auxiliary conditions, such as initial and terminal conditions[13], can affect the stability and convergence of the method, as further discussed in [15]. A key question is whether the regularization effect from neural network approximations could help alleviate these issues.

4. Convergence Analysis

In this section, we first present the approach of using neural networks to approximate the linear system derived from the discretized LMMs scheme (3.7), as well as the linear system obtained after incorporating auxiliary conditions. We then analyze the relationship between the error of the learned governing function f_{NN} and the original governing function f, which consists of both the discretization error of the numerical scheme and the network approximation error. Finally, we trace the error in the governing function back to the error in the original state function x.

4.1. Network-based discovery method

Let us revisit the dynamics discovery on a single trajectory, as introduced in Section 3. Conventional LMMs are straightforward to implement, yielding solutions by solving a linear system. However, this approach only computes the governing function f at fixed, equidistant time steps, leaving the relationship between f and the state x unknown. One way to address this is to approximate each

component of f with structured functions, such as neural networks, polynomials, or splines allowing closed-form expressions for f through optimization. Once f is explicitly recovered, future states x on the same or nearby trajectories can be predicted by solving (3.1) with initial or perturbed conditions.

Neural networks are especially effective for this purpose, particularly when d is moderately large, as they handle high-dimensional inputs better than other structures. Thus, we focus on network-based methods here, though the approach generalizes to other types of approximations.

We consider neural network approximations based on the LMMs scheme (3.6). Generally, We use $\mathcal{NN}(k,G,N)$ to denote the set of all two-layer B-spline KANs with k-degree, G nodes, and N represents the number of basis functions in second layer. Now we introduce a network $f_{\mathcal{NN}}(\boldsymbol{x}) \in \mathcal{NN}(k,G,N)$ to approximate $f(\boldsymbol{x})$ an arbitrary component of $f(\cdot)$. The neural network method can be developed by

$$h \sum_{m=0}^{M} \beta_m f_{NN}(\boldsymbol{x}_{n-m}) = \sum_{m=0}^{M} \alpha_m \boldsymbol{x}_{n-m}, \quad n = M, M+1, \dots, N_1,$$
(4.1)

where \boldsymbol{x}_n for $n = 0, \dots, N_1$ are given sample locations.

Unfortunately, if the VC-dimension of $\mathcal{NN}(k, G, N)$ is very small, then there does not exist an $f_{\mathcal{NN}} \in \mathcal{NN}(k, G, N)$ that can exactly satisfy (4.1). Therefore, we typically seek $f_{\mathcal{NN}} \in \mathcal{NN}(k, G, N)$ by minimizing the residual of (4.1) within a machine learning framework, such that

$$J_h(f_{\mathcal{N}\mathcal{N}}) = \min_{u \in \mathcal{N}\mathcal{N}(k,G,N)} J_h(u), \tag{4.2}$$

where

$$J_h(u) := \frac{1}{N_1 - M + 1} \sum_{n=M}^{N_1} \left| \sum_{m=0}^{M} \beta_m u(\boldsymbol{x}_{n-m}) - \sum_{m=0}^{M} h^{-1} \alpha_m x_{n-m} \right|^2.$$
 (4.3)

However, similar to the underdetermined linear system (3.7) which has an infinite number of solutions, ensuring a unique minimizer at the grid points necessitates additional conditions. The conditions are introduced and an augmented loss function is constructed based on equation (4.3) to fulfill the objective, as detailed in the work [9].

The auxiliary minimization problem is formulated to find the optimal neural network function $f_{\mathcal{N}\mathcal{N}}$ within the space of neural networks $\mathcal{N}\mathcal{N}(k,G,N)$, as depicted in Equation (4.4):

$$J_{a,h}(f_{\mathcal{N}\mathcal{N}}) = \min_{u \in \mathcal{N}\mathcal{N}(k,G,N)} J_{a,h}(u). \tag{4.4}$$

The objective function $J_{a,h}(u)$ is defined as the sum of squared errors over a training dataset, which includes both the approximation error and the collocation error, as shown in Equation (4.5):

$$J_{a,h}(u) := \frac{1}{\tau} \left(\sum_{n=s}^{s+N_a-1} \left| u(\boldsymbol{x}_n) - \frac{1}{h} \sum_{m=0}^{p} \mu_m x_{n+m} \right|^2 + \sum_{n=M}^{N_1} \left| \sum_{m=0}^{M} \beta_m u(\boldsymbol{x}_{n-m}) - \sum_{m=0}^{M} h^{-1} \alpha_m x_{n-m} \right|^2 \right). \tag{4.5}$$

This enhanced optimization strategy ensures the uniqueness of the minimizer at the grid points within the function space. For further details on the network implementation process, refer to Figure 4.1.

4.2. Error estimates

We consider the error estimation of the discovery on the trajectory $\Delta := \{x(t) : 0 \le t \le T\}$.

KAN(x;c)

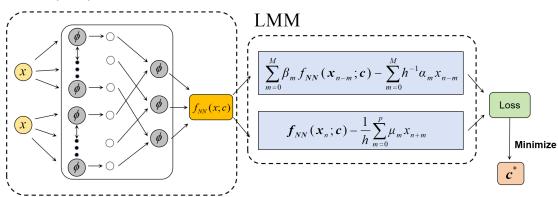


Figure 4.1. Schematic representation of the network architecture employing LMMs in conjunction with KANs for dynamical system discovery.

Definition 4.1 (l^2 seminorm [9]). Let $\Delta := \{x(t) : 0 \le t \le T\}$, for any $f \in C(\Delta)$ with a given h > 0, the l^2 seminorm as

$$|f|_{2,h} := \left(\frac{1}{N_1 + 1} \sum_{n=0}^{N_1} |f(\boldsymbol{x}_n)|^2\right)^{1/2}.$$
 (4.6)

As discussed above, for a specific LMM, some states in $\{x_n\}_{n=0}^{N_1}$ may not be involved in the scheme. For fairness, we study the convergence at all involved states $\{x_n\}_{n=r}^q$. Therefore, we rewrite $|f|_{2,h}$ as the LMM-related seminorm

$$|f|_{2,h} := \left(\frac{1}{\tau} \sum_{n=r}^q |f(\boldsymbol{x}_n)|^2\right)^{1/2}$$
 for all $f \in C(\Delta)$.

Next, we represent $\{f(\boldsymbol{x}_n)\}_{n=r}^q$ as the vector $\vec{\boldsymbol{f}} := [f(\boldsymbol{x}_r) \ f(\boldsymbol{x}_{r+1}) \ \dots \ f(\boldsymbol{x}_q)]^T$ and aim to estimate the distance between $f_{\mathcal{N}\mathcal{N}}$ and f. Here, $f_{\mathcal{N}\mathcal{N}}$ denotes the minimizer of (4.4), the result of our network optimization. To evaluate these errors, we account for both the numerical discretization error and the network approximation error, and we present their relationship below.

Lemma 4.1 (Theorem 5.1 of [9]). In the dynamical system (3.1), suppose $\mathbf{x}(t) \in C^{\infty}([0,T])^d$ and \mathbf{f} is defined in Δ' , a small neighborhood of Δ . Let f be an arbitrary component of \mathbf{f} and $h := T/N_1$ with an integer $N_1 > 0$, then

$$|f_{\mathcal{A}} - f|_{2,h} < C\kappa_2(\mathbf{A}_h) \left(h^p + e_{\mathcal{A}}\right), \tag{4.7}$$

where $f_{\mathcal{A}} \in \mathcal{A}$ is a global minimizer of $J_{a,h}$ defined by (4.5) corresponding to an LMM with order p and $\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2$.

Lemma 4.2 (Theorem 5.4 of [9]). Let A_h be the matrix defined by (3.9), and $p_h(z)$ be the following polynomial

$$p_h(z) = \sum_{i=N-q}^{M-r} \beta_i z^{M-r-i}.$$

If all roots of $p_h(z)$ have modulus smaller than 1, then $\kappa_2(\mathbf{A}_h)$ is uniformly bounded with respect to N_1 .

Theorem 4.1. In the dynamical system (3.1), suppose $\mathbf{x}(t) \in C^{\infty}([0,T])^d$ and \mathbf{f} is defined in Δ' , a small neighborhood of Δ . Let f be an arbitrary component of \mathbf{f} . Also, let $N_1 > 0$ be an integer and $h := T/N_1$, then we have

$$|f_{\mathcal{N}\mathcal{N}} - f|_{2,h} < C\kappa_2(\mathbf{A}_h) \left(h^p + e_{\mathcal{N}\mathcal{N}}(k, G, N) \right), \tag{4.8}$$

with $e_{\mathcal{NN}}(k,G,N) = 2R_{\Delta'}N(L_Gd+1) \cdot \omega_f\left(\min\left\{\frac{1}{\sqrt{2k-2}},\sqrt{\frac{k}{12}}\cdot\frac{1}{G}\right\}\right)$, where $f_{\mathcal{NN}} \in \mathcal{NN}(k,G,N)$ is a global minimizer of $J_{a,h}$ defined by (4.5) corresponding to an LMM with order p; C is a constant independent of h,k,G,N. In particular, if $\kappa_2(\mathbf{A}_h)$ is uniformly bounded for all h>0, then

$$\lim_{k,G,N\to\infty,h\to 0} |f_{\mathcal{N}\mathcal{N}} - f|_{2,h} = 0.$$

Proof. According to Lemma 4.1, if we replace the function approximation set with $\mathcal{NN}(k,G,N)$, we obtain the following expression

$$|f_{\mathcal{N}\mathcal{N}} - f|_{2,h} < C\kappa_2(\mathbf{A}_h) (h^p + e_{\mathcal{N}\mathcal{N}}(k, G, N)).$$

By applying Corollary 2.1, we substitute $e_{\mathcal{A}}$ with the error $e_{\mathcal{N}\mathcal{N}}(k,G,N) = 2R_{\Delta'}N(L_Gd+1)$ $\omega_f\left(\min\left\{\frac{1}{\sqrt{2k-2}},\sqrt{\frac{k}{12}}\cdot\frac{1}{G}\right\}\right)$ of our two-layer B-spline KANs, where $R_{\Delta'}$ denotes the radius of the domain of the function we aim to approximate. Thus, the total error is given by

$$|f_{\mathcal{N}\mathcal{N}} - f|_{2,h} < C\kappa_2(\boldsymbol{A}_h) \left(h^p + 2R_{\Delta'}N(L_Gd + 1) \cdot \omega_f \left(\min\left\{ \frac{1}{\sqrt{2k-2}}, \sqrt{\frac{k}{12}} \cdot \frac{1}{G} \right\} \right) \right).$$

where uniform boundedness of $\kappa_2(\mathbf{A}_h)$ here is guaranteed by Lemma 4.2.

Now that we have derived the approximation error of $f_{\mathcal{N}\mathcal{N}}$ for approximating f, we proceed to analyze the error relationship between $x_{\mathcal{N}\mathcal{N}}$, obtained by solving with the approximated $f_{\mathcal{N}\mathcal{N}}$, and the exact solution x derived from the original function.

Theorem 4.2. Suppose $\mathbf{x}(t), \mathbf{x}_{\mathcal{N}\mathcal{N}}(t) \in C^{\infty}([0,T])^d$ and $\mathbf{f}, \mathbf{f}_{\mathcal{N}\mathcal{N}}$ is defined in Δ' , a small neighborhood of Δ , in the dynamical system (3.1), If the error between \mathbf{f} and $\mathbf{f}_{\mathcal{N}\mathcal{N}}$ is $\boldsymbol{\varepsilon}$, then the corresponding error between the solutions $\mathbf{x}(t)$ and $\mathbf{x}_{\mathcal{N}\mathcal{N}}(t)$ is given by

$$\|\boldsymbol{x}(t) - \boldsymbol{x}_{\mathcal{N}\mathcal{N}}(t)\| \le \left(\|\boldsymbol{x}(0) - \boldsymbol{x}_{\mathcal{N}\mathcal{N}}(0)\| + \varepsilon t\right) e^{L_f t}.$$
(4.9)

Proof. Let $z(t) = x(t) - x_{NN}(t)$,

$$\begin{split} \frac{d\boldsymbol{z}(t)}{dt} &= \frac{d\boldsymbol{x}(t)}{dt} - \frac{d\boldsymbol{x}_{\mathcal{N}\mathcal{N}}(t)}{dt} \\ &= \boldsymbol{f}(t, \boldsymbol{x}(t)) - \boldsymbol{f}_{\mathcal{N}\mathcal{N}}(t, \boldsymbol{x}_{\mathcal{N}\mathcal{N}}(t))) \\ &= \boldsymbol{f}(t, \boldsymbol{x}(t)) - (\boldsymbol{f}(t, \boldsymbol{x}_{\mathcal{N}\mathcal{N}}(t)) + \boldsymbol{\varepsilon}). \end{split}$$

Integrating both sides of the above equation yields

$$z(t) = z(0) + \int_0^t \left(f(s, x(s)) - f(s, x_{\mathcal{N}\mathcal{N}}(s)) \right) ds + \int_0^t \varepsilon ds$$
$$= (z(0) + \varepsilon t) + \int_0^t \left(f(s, x(s)) - f(s, x_{\mathcal{N}\mathcal{N}}(s)) \right) ds.$$

Taking norms on both sides of the above equation, one has

$$\|\boldsymbol{z}(t)\| \le \|\boldsymbol{z}(0) + \boldsymbol{\varepsilon}t\| + \left\| \int_0^t \left(\boldsymbol{f}(s, \boldsymbol{x}(s)) - \boldsymbol{f}(s, \boldsymbol{x}_{\mathcal{N}\mathcal{N}}(s)) \right) ds \right\|$$

$$\le \|\boldsymbol{z}(0) + \boldsymbol{\varepsilon}t\| + L_f \int_0^t \|\boldsymbol{z}(s)\| \, ds.$$

It follows from Gronwall's inequality that

$$\|\boldsymbol{z}(t)\| \le \|\boldsymbol{z}(0) + \boldsymbol{\varepsilon}t\|e^{L_f t},$$

which implies (4.9). This completes the proof.

5. Numerical experiments

This section presents a series of numerical experiments to substantiate our theoretical findings on uncovering hidden dynamics utilizing Linear Multistep Methods (LMMs) in conjunction with Kolmogorov-Arnold Networks (KANs). Drawing on precedents set by [9, 15, 26], we examine three distinct examples: a simple linear ODE system, a glycolytic oscillator, and a model of opinion dynamics.

We assess the error orders of Adams-Bashforth (AB), Backward Differentiation Formula (BDF), and Adams-Moulton (AM) schemes within this context, with detailed coefficients referenced in [7]. The experimental framework is outlined as follows:

- Optimizer and Hyperparameters: The optimization of network-based models is efficiently conducted using the Adam subroutine, accessible within the PyTorch library, which implements the algorithm detailed in [16]. Batch gradient descent is applied with a learning rate of 0.01.
- Network Configuration: We engage two-layer KANs with B-spline activation functions for approximation purposes, initializing weights according to the Xavier uniform distribution as described in [12]. Guided by the network error component within our theoretical results, we experiment with parameters k and G to identify their effective values.
- Data Generation: In subsequent examples, data generation is performed by solving dynamical systems numerically with scipy.integrate.odeint, which automatically switches between Adams methods for non-stiff problems and BDF for stiff problems. The solver uses rtol = 10⁻¹³ and atol = 10⁻¹³ to ensure high precision. By basing our dynamical discoveries on such precisely generated data, we can more accurately assess the validity and reliability of the vector field functions obtained through our fitting process.

5.1. Linear ode system

Let us consider the following model problem

$$\begin{cases} \frac{dx}{dt} = 2x + 3y, \\ \frac{dy}{dt} = -4y, \end{cases}$$
(5.1)

with the initial state $[x_0, y_0]^T = [0, 1]^T$. The state of the system is explicitly given by $x(t) = \frac{1}{2}e^{2t} - \frac{1}{2}e^{-4t}$ and $y(t) = e^{-4t}$.

Firstly, we design an experiment to investigate the approximation capabilities of various multistep methods under identical network conditions. We systematically compare their performance, with

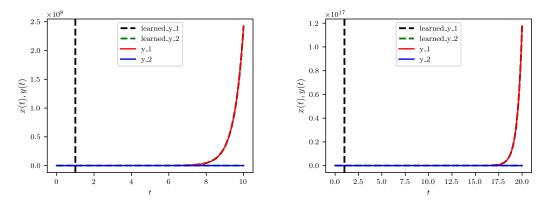


Figure 5.1. Left panel: The black dashed vertical line at t = 1 demarcates the boundary between the training interval [0,1] and the prediction interval [1,10]. Right panel: The black dashed vertical line at t = 1 demarcates the boundary between the training interval [0,1] and the extended prediction interval [1,20].

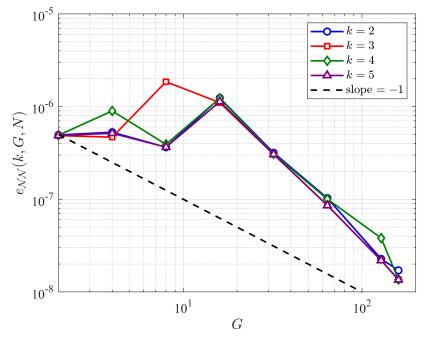


Figure 5.2. Influence of B-Spline KANs parameters: $e_{\mathcal{N}\mathcal{N}}(k,G,N)$ as a function of degree k and node number G.

numerical results detailed in Table 1. Notably, the second-order AM 1-step method outperforms others, achieving the lowest trajectory error and demonstrating superior stability, corroborating findings from [9, 26]. Subsequently, we focus on the impact of parameters k and G of the B-spline basis functions on the KAN network error $e_{\mathcal{N}\mathcal{N}}(k,G,N)$ under the AM 1-step scheme. Experiments to generate Figure 5.2 are conducted with B-spline degrees k=2,3,4,5 and number of nodes G=4,8,16,32,64,128,160 over the interval [0,10] with a step size h=0.001 for the AM scheme. The results demonstrate that when k is relatively small, the parameter G predominantly influences the error $e_{\mathcal{N}\mathcal{N}}(k,G,N)$. Figure 5.2 demonstrates that with the increase of G towards infinity, the convergence rate of the network $f_{\mathcal{N}\mathcal{N}}$ approximating the vector field f with respect to 1/G exceeds 1.

Table 5.1. Error between f and f_{NN} for multistep methods with varying step numbers M using a two-layer B-Spline KANs architecture.

M-step	1-step	2-step	3-step	4-step	5-step	6-step
AB	8.71e-08	3.46e-07	7.80e-07	1.39e-06	2.16e-06	3.12e-06
AM	8.60e-08	3.44e-07	7.75e-07	2.15e-06	2.16e-06	3.11e-06
BDF	8.61e-08	1.54e-07	2.32e-07	3.20e-07	4.16e-07	5.21e-07

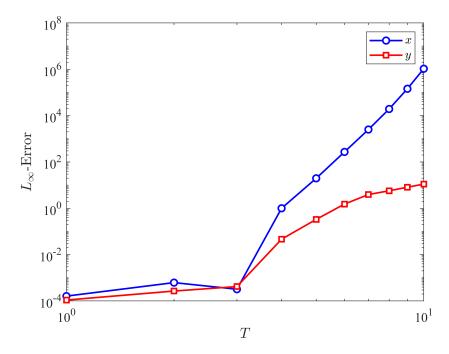


Figure 5.3. The L^{∞} -norm error between $\boldsymbol{x}(t)$ and $\boldsymbol{x}_{\mathcal{N}\mathcal{N}}(t)$ over the interval [0,T].

Next, we examine the Theorem 4.2. In our numerical experiment, we utilize the previously determined network configuration, specified by the parameters k = 3, G = 64, $h = 10^{-3}$, and 2200 iterations with the AM 1-step method. We calculate the errors between the neural network approximation $\boldsymbol{x}_{\mathcal{NN}}$ and the original function \boldsymbol{x} for T = 1, 2, ..., 10. These results are depicted in Figure 5.3 using a log-log plot, showing that the errors for both solutions \boldsymbol{x} and \boldsymbol{y} increase

exponentially with T, consistent with our theoretical predictions.

5.2. Glycolytic oscillator

This subsection presents a simulation of the glycolytic oscillator model, exemplifying the complex nonlinear dynamics inherent in biological systems. The model is formulated by a system of ordinary differential equations that describe the concentrations of various species:

$$\begin{cases}
\frac{dS_1}{dt} = J_0 - \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q}, \\
\frac{dS_2}{dt} = 2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5, \\
\frac{dS_3}{dt} = k_2 S_2 (N - S_5) - k_3 S_3 (N - S_6), \\
\frac{dS_4}{dt} = k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7), \\
\frac{dS_5}{dt} = k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5, \\
\frac{dS_6}{dt} = -2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} + 2k_3 S_3 (A - S_6) - k_5 S_6, \\
\frac{dS_7}{dt} = \psi \kappa (S_4 - S_7) - k_7 S_7
\end{cases} \tag{5.2}$$

with initial condition $S_0 = [1.125, 0.95, 0.075, 0.16, 0.265, 0.7, 0.092]$. Model parameters are detailed in Table 5.2.

k_1	k_2	k_3	k_4	k_5	k_6	k_7
100.0	6.0	16.0	100.0	1.28	12.0	1.8
J_0	κ	q	K_1	ψ	N	A
2.5	13.0	4	0.52	0.1	1.0	4.0

Table 5.2. Parameters of the glycolytic oscillator

Employing the same network configuration as in the previous example (k = 3, G = 64, $h = 10^{-3}$, 2200 iterations, AM 1-step), we calculate the errors between $\boldsymbol{x}_{\mathcal{N}\mathcal{N}}$ and the original function over the interval [0,15]. As shown in Figure 5.4, the learned system accurately captures the dynamics' form. We observe that for high-dimensional cases, our error remains small over an extended period, indicating the stability and reliability of our approach.

5.3. Opinion dynamics

This numerical experiment delves into the opinion dynamics model, a typical example within the realm of self-organized dynamical systems. The model is governed by the following system of equations:

$$\frac{d}{dt}\mathbf{x}_i = \alpha \sum_{j \neq i} a_{ij}(\mathbf{x}_j - \mathbf{x}_i), \quad a_{ij} = \frac{\phi_{ij}}{\sum_k \phi_{ik}}, \quad \phi_{ij} := \phi(|\mathbf{x}_j - \mathbf{x}_i|). \tag{5.3}$$

Here, $\alpha > 0$ is a scaling parameter, and ϕ is a scaled influence function defined as the characteristic function $\phi = 1\chi_{[0,1]}$, which operates on the "difference of opinions," denoted by $|\mathbf{x}_j - \mathbf{x}_i|$. The metric $|\cdot|$ signifies the absolute value.

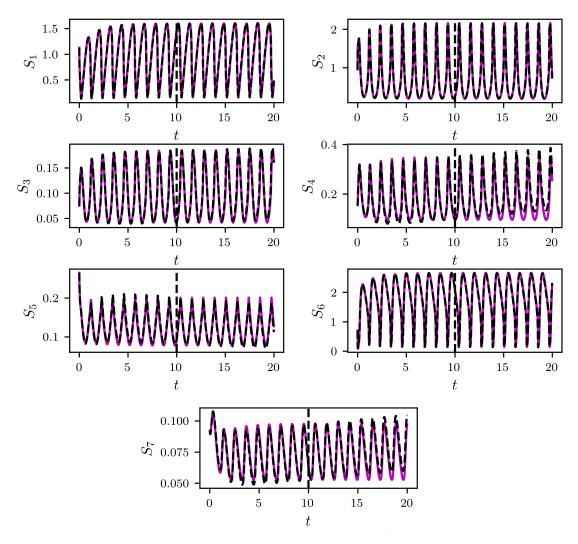


Figure 5.4. Comparison of trajectories for the Gylcolytic Oscillator: exact vs. learned dynamics. Solid blue lines indicate the exact dynamics, whereas dashed black lines represent the learned dynamics. The black dashed vertical line at t = 10 demarcates the boundary between the 'training' interval [0, 10] and the 'prediction' interval [10, 20].

We examine the system's behavior under random initial conditions across dimensions d = 50, 100, 200, and 400. The parameters for the KAN are set to k = 3, G = 64, and $h = 10^{-3}$, with 3000 iterations per simulation. The Adams-Moulton method of order 1 is employed in conjunction with the KAN setup, with parameter arrays customized for different dimensions: [50 101 50], [100 201 100], [200 401 200], and [400 801 400].

A pivotal aspect of this study is the interplay between the KANs' approximation capability and the system's dimensionality d. Our theoretical analysis indicates that the lower bound of the error, as quantified by the VC dimension, escalates with d but ultimately converges to a constant C. This convergence suggests that the KAN model's approximation error is contained even amidst escalating system dimensionality, thereby ensuring the model's efficacy in high-dimensional scenarios. This

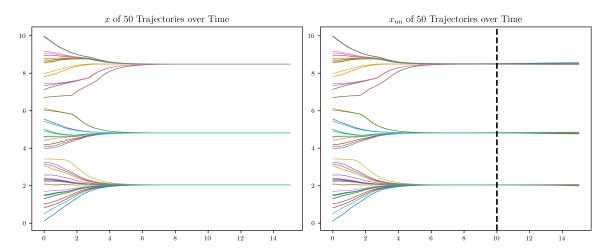


Figure 5.5. A comparison of the temporal evolution of opinion dynamics as described by (5.3), solved using a KAN, with the actual evolution. (Left figure: Actual evolution; Right figure: Network solutions.) The black dashed vertical line at t = 10 demarcates the boundary between the 'training' interval [0, 10] and the 'prediction' interval [10, 15].

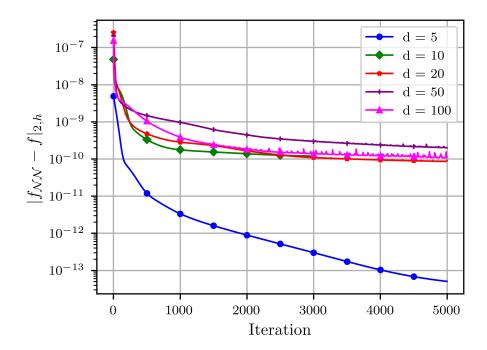


Figure 5.6. Training Error between \boldsymbol{f} and $\boldsymbol{f}_{\mathcal{N}\mathcal{N}}$ of Opinion Dynamics Model at T=10.

Table 5.3. The L^{∞} -norm error between the true state $\boldsymbol{x}(t)$ and the approximation $\boldsymbol{x}_{\mathcal{N}\mathcal{N}}(t)$ across the interval [0, 10].

d	50	100	200	400
Errors	3.92960581e-02	1.62031044e-01	1.77045530e-01	1.71368496e-01

attribute is indispensable for applications involving high-dimensional data, as it ascertains that the KAN model sustains a consistent approximation error margin. To substantiate these theoretical insights, we conduct a comprehensive error analysis for dimensions d = 50, 100, 200, and 400, as detailed in Table 5.3. Figure 5.5 visually corroborates the algorithm's proficiency in approximating the 50-dimensional system over the interval [0, 10] by contrasting it with the original orbit plot.

The graph in Figure 5.6 illustrates the convergence behavior of the error bounds for different dimensions d in the context of KANs. The error's lower bound is derived from the VC dimension. Conversely, the upper bound reflects the network's ability to approximate the target function.

For lower dimensions (d=5), the graph shows a rapid decrease in error as the network undergoes optimization, suggesting a strong approximation capability. As the dimension d increases, the error's lower bound eventually approaching a fixed upper limit. This behavior is observed in the graph, where the error for higher dimensions (d=10,20,50,100) stabilizes at a nearly constant value.

References

- [1] R. P. Agarwal. Difference Equations and Inequalities: Theory, Methods, and Applications. CRC Press, 1 edition, 2000.
- [2] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. J. Mach. Learn. Res., 20(63):1–17, 2019.
- [3] G. Benettin and A. Giorgilli. On the Hamiltonian interpolation of near-to-the identity symplectic mappings with application to symplectic integration algorithms. *J. Stat. Phys.*, 74(5-6):1117–1143, 1994.
- [4] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. Proc. Nat. Acad. Sci. India Sect. A, 104(24):9943–9948, 2007.
- [5] J. Braun and M. Griebel. On a constructive proof of kolmogorov's superposition theorem. Constr. Approx., 30(3):653–675, 2009.
- [6] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Nat. Acad. Sci. India Sect. A*, 113(15):3932–3937, 2016.
- [7] R. D'Ambrosio. Numerical Approximation of Ordinary Differential Problems: From Deterministic to Stochastic Numerical Methods. Springer, 2023.
- [8] C. de Boor. A Practical Guide to Splines. Springer, New York, 1978.
- [9] Q. Du, Y. Gu, H. Yang, and C. Zhou. The discovery of dynamics via linear multistep methods and deep learning: Error estimation. SIAM J. Numer. Anal., 60(4):2014–2045, 2022.
- [10] W. E. A Proposal on Machine Learning via Dynamical Systems. Commun. Math. Stat., 5(1):1–11, 2017.
- [11] J. Feng, C. Kulick, Y. Ren, and S. Tang. Learning particle swarming models from data with Gaussian processes. *Math. Comp.*, 93(349):2391–2437, 2024.

- [12] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. J. Mach. Learn. Res., 9:249–256, 2010.
- [13] N. S. Gulgec, Z. Shi, N. Deshmukh, S. Pakzad, and M. Takáč. FD-Net with Auxiliary Time Steps: Fast Prediction of PDEs using Hessian-Free Trust-Region Methods, 2019.
- [14] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. Science, 349(6245):255–260, 2015.
- [15] R. T. Keller and Q. Du. Discovery of dynamics using linear multistep methods. SIAM J. Numer. Anal., 59(1):429–455, 2021.
- [16] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2014.
- [17] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. *Math. Comput. Model. Dyn. Syst.*, 11(4):411–424, 2005.
- [18] V. Kůrková. Kolmogorov's Theorem Is Relevant. Neural Comput., 3(4):617-622, 1991.
- [19] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark. KAN: Kolmogorov-Arnold Networks, 2024.
- [20] Z. Long, Y. Lu, and B. Dong. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. J. Comput. Phys., 399:108925, 2019.
- [21] F. Lu, M. Zhong, S. Tang, and M. Maggioni. Nonparametric inference of interaction laws in systems of agents from trajectory data. *Proc. Nat. Acad. Sci. India Sect. A*, 116(29):14424–14433, 2019.
- [22] J. Lu, Z. Shen, H. Yang, and S. Zhang. Deep Network Approximation for Smooth Functions. SIAM J. Math. Anal., 53(5):5465–5506, 2021.
- [23] T. Poggio, A. Banburski, and Q. Liao. Theoretical issues in deep networks. Proc. Nat. Acad. Sci. India Sect. A, 117(48):30039–30045, 2020.
- [24] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *J. Comput. Phys.*, 335:736–746, 2017.
- [25] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Machine learning of linear differential equations using Gaussian processes. *J. Comput. Phys.*, 348:683–693, 2017.
- [26] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems, 2018.
- [27] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Data-driven discovery of partial differential equations. *Science*, 3(4):e1602614, 2017.
- [28] S. H. Rudy, J. Nathan Kutz, and S. L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *J. Comput. Phys.*, 396:483–506, 2019.
- [29] M. Schmidt and H. Lipson. Distilling Free-Form Natural Laws from Experimental Data. Science, 324(5923):81–85, 2009.

- [30] Z. Shen, H. Yang, and S. Zhang. Deep Network Approximation Characterized by Number of Neurons. *Commun. Comput. Phys.*, 28(5):1768–1811, 2020.
- [31] Z. Shen, H. Yang, and S. Zhang. Deep Network With Approximation Error Being Reciprocal of Width to Power of Square Root of Depth. *Neural Comput.*, 33(4):1005–1036, 2021.
- [32] Z. Shen, H. Yang, and S. Zhang. Neural network approximation: Three hidden layers are enough. Neural Netw., 141:160–173, 2021.
- [33] Z. Shen, S. Zhang, and H. Yang. Optimal approximation rate of ReLU networks in terms of width and depth. *J. Math. Pures Appl.*, 157:101–135, 2022.
- [34] J. A. Thie. Modern Control Systems. Nuclear Science and Engineering, 31(3):560-560, 1968.
- [35] R. Tipireddy, P. Perdikaris, P. Stinis, and A. Tartakovsky. A comparative study of physicsinformed neural network models for learning unknown dynamics and constitutive relations, 2019.
- [36] X. Xie, G. Zhang, and C. G. Webster. Non-Intrusive Inference Reduced Order Model for Fluids Using Deep Multistep Neural Network. *Mathematics*, 7(8):757, 2019.
- [37] C. J. Zarowski. An Introduction to Numerical Analysis for Electrical and Computer Engineers. Wiley, 2004.
- [38] S. Zhang and G. Lin. Robust data-driven discovery of governing physical laws with error bars. *Proc. Roy. Soc. Edinburgh Sect. A*, 474(2217):20180305, 2018.
- [39] S. Zhang and G. Lin. SubTSBR to tackle high noise and outliers for data-driven discovery of differential equations. J. Comput. Phys., 428:109962, 2021.
- [40] M. Zhong, J. Miller, and M. Maggioni. Data-driven discovery of emergent behaviors in collective dynamics. Phys. D, 411:132542, 2020.