From Fourier to Neural ODEs: Flow Matching for Modeling Complex Systems

Xin Li [1] **Jingdong Zhang** [2 3] **Qunxi Zhu** [3] **Chengli Zhao** [1] **Xue Zhang** [1] **Xiaojun Duan** [1] **Wei Lin** [2 3 4 5]

Abstract

Modeling complex systems using standard neural ordinary differential equations (NODEs) often faces some essential challenges, including high computational costs and susceptibility to local optima. To address these challenges, we propose a simulation-free framework, called Fourier NODEs (FNODEs), that effectively trains NODEs by directly matching the target vector field based on Fourier analysis. Specifically, we employ the Fourier analysis to estimate temporal and potential high-order spatial gradients from noisy observational data. We then incorporate the estimated spatial gradients as additional inputs to a neural network. Furthermore, we utilize the estimated temporal gradient as the optimization objective for the output of the neural network. Later, the trained neural network generates more data points through an ODE solver without participating in the computational graph, facilitating more accurate estimations of gradients based on Fourier analysis. These two steps form a positive feedback loop, enabling accurate dynamics modeling in our framework. Consequently, our approach outperforms state-of-the-art methods in terms of training time, dynamics prediction, and robustness. Finally, we demonstrate the superior performance of our framework using a number of representative complex systems.

[1]College of Science, National University of Defense Technology, Changsha, Hunan 410073, China. [2]School of Mathematical Sciences, LMNS, and SCMS, Fudan University, China. [3]Research Institute of Intelligent Complex Systems, Fudan University, China. [4]State Key Laboratory of Medical Neurobiology and MOE Frontiers Center for Brain Science, Institutes of Brain Science, Fudan University, China. [5]Shanghai Artificial Intelligence Laboratory, China. Correspondence to: Qunxi Zhu <qxzhu16@fudan.edu.cn>, Chengli Zhao <chenglizhao@nudt.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Complex dynamical systems have garnered considerable attention across various disciplines in the natural and social sciences. These systems are typically characterized by ordinary differential equations (ODEs) or partial differential equations (PDEs) (Perko, 1991; Meiss, 2007; Li et al., 2024). However, the explicit forms of these underlying systems are often completely unknown *a priori*, and the available information about such systems is commonly obtained through experimentally collected time series data. Unfortunately, these data are often noisy and limited due to measurement errors, resource constraints, and the high cost of data collection. As a result, there is an urgent need to uncover the underlying dynamics of these systems without prior knowledge of the specific governing equations, relying solely on the measured time series data (Legaard et al., 2021; Zhu et al., 2023). The successful modeling of these systems enables downstream tasks such as prediction (Carroll, 2018; Pathak et al., 2018) and control (Sanhedrai et al., 2020; Zhang & Zhou, 2022). Therefore, the development of an efficient and robust data-driven approach for system modeling is of utmost significance.

Recently, there has been a tremendous amount of interest in applying machine learning technologies, such as autoregressive models (Ding et al., 2018), sparsity-promoting methods (Kaiser et al., 2017; Meiss, 2007), reservoir computing (Pathak et al., 2018; Li et al., 2023), and Neural ODEs (NODEs) (Chen et al., 2018), to model complex dynamical systems. In particular, NODEs and their extensions (Holt et al., 2022; Bilovs et al., 2021; Lanzieri et al., 2022), which involve continuously defined dynamics, have been extensively utilized for coping with the continuous-time datasets due to the direct parameterization of the vector field of ODEs, enabling them to naturally capture the underlying dynamics of continuous-time systems solely based on observational data. Despite their successes, NODEs often encounter challenges related to high computational costs and susceptibility to local optima. During the training process, the NODEs may gradually exhibit complex dynamic behaviors, such as drastic fluctuations or stiffness. These behaviors significantly increase the time required for numerical solving and the memory consumption for backpropagation (Chen et al., 2018; Kim et al., 2021; Finlay et al., 2020). Furthermore, the corresponding adjoint method for

computing gradients is numerically sensitive (Zhuang et al., 2020).

To address the challenges mentioned above and enhance the speed and robustness of dynamic modeling, we propose a framework called Fourier NODEs (FNODEs). The key advantages of our proposed framework are summarized as follows.

- The utilization of Fourier analysis theory provides theoretical guarantees for accurately estimating the gradient flows of dynamical systems with limited and noisy data.
- Directly matching the estimated gradient flows, serving as a simulation-free framework, allows the FNODEs to significantly reduce the training time by more than tenfold compared to the standard NODEs.
- The data augmentation training strategy is introduced by generating more data points for better estimating the gradient flows and then retraining the model, which forms a positive feedback loop, resulting in improved robustness and accuracy in dynamic modeling.
- Our framework can be easily applied for resolution-invariant modeling of PDEs by additionally incorporating the estimated higher-order spatial gradients based on Fourier analysis into the model, expanding the scopes of classical NODEs.

2. Related Work

NODEs (Chen et al., 2018), a class of continuous-depth neural networks, have garnered substantial interest in recent years due to their pivotal role across various scientific disciplines. The typical residual block, represented as

$$\boldsymbol{z}(t+1) = \boldsymbol{z}(t) + \boldsymbol{F}_{\boldsymbol{\theta}}[\boldsymbol{z}(t), t],$$

can be construed as an Euler discretization (with a time step $\Delta t = 1$) of an ODE (Chen et al., 2018),

$$\dot{\boldsymbol{z}} = \boldsymbol{F}_{\boldsymbol{\theta}}[\boldsymbol{z}(t), t],$$

where $\boldsymbol{z}(t) \in \mathbb{R}^d$ denotes the state of the d-dimensional (-D) system, t is the time, also interpreted as a continuous "depth" in the residual block, and $\boldsymbol{F}_{\boldsymbol{\theta}} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ can be any neural network architecture with the parameter vector $\boldsymbol{\theta}$ to be optimized. Consequently, given the initial value $\boldsymbol{z}(t_0)$, the state at any time t_1 can be computed using an ODE solver:

$$\begin{aligned}\boldsymbol{z}(t_1) &= \text{ODEsolve}[\boldsymbol{z}(t_0), \boldsymbol{F}_{\boldsymbol{\theta}}, t_0, t_1] \\ &= \boldsymbol{z}(t_0) + \int_{t_0}^{t_1} \boldsymbol{F}_{\boldsymbol{\theta}}[\boldsymbol{z}(t), t]\mathrm{d}t.\end{aligned}$$

The successful application of NODEs in modeling continuous time data has led to the emergence of numerous works aimed at addressing the limitations of the original NODEs. These include augmented NODEs (Dupont et al., 2019), neural delay differential equations (Zhu et al., 2021), neural controlled differential equations (Kidger et al., 2020), neural stochastic differential equations (Liu et al., 2019), and neural integro-differential equations (Zappala et al., 2022). Despite the effectiveness of these methods in handling specific types of dynamical systems, they are not exempt from the challenges of high computational cost during the training process and difficulties in converging to the global optimal solution.

3. The Framework of FNODEs

To establish the Fourier NODEs (FNODEs) framework, we consider a controlled dynamical system of the following general form,

$$\dot{\boldsymbol{s}} = \boldsymbol{f}[\boldsymbol{s}(t), \boldsymbol{u}(t)], \tag{1}$$

where $\boldsymbol{s}(t) \in \mathbb{R}^d$ represents the state of the d-D system, $\boldsymbol{u}(t) \in \mathbb{R}^m$ is a time-dependent controller, and $\boldsymbol{f} : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^d$ embodies the inherent dynamics of the system. Typically, the dynamical system is often unknown. Consequently, a common scenario is to describe the underlying dynamics via learning a surrogate model $\boldsymbol{F}_\theta$ parameterized by a neural network solely based on the observational data. In this work, we assume that the controller $\boldsymbol{u}(t)$ is known, and sampled from a specific family of functions.

The approximation of temporal gradients. Let $\boldsymbol{h}(t)$ be a continuously differentiable vector-valued function that is T-periodic. From a spectral approximation perspective, we can express $\boldsymbol{h}(t)$ as a Fourier series expansion:

$$\boldsymbol{h}(t) = \lim_{K\to\infty} \boldsymbol{H}_K(t) = \lim_{K\to\infty} \left\{ \frac{\boldsymbol{a}_0}{2} + \sum_{k=1}^{K} \left[\boldsymbol{a}_k \sin\left(\frac{2\pi}{T}kt\right) + \boldsymbol{b}_k \cos\left(\frac{2\pi}{T}kt\right) \right] \right\},$$

where $\boldsymbol{H}_K(t)$ represents the K-th partial sum of the Fourier series, $\boldsymbol{a}_k = 2\left[\int_0^T \boldsymbol{h}(t)\cos(kt)\mathrm{d}t\right]/T$, and $\boldsymbol{b}_k = 2\left[\int_0^T \boldsymbol{h}(t)\sin(kt)\mathrm{d}t\right]/T$. We can then utilize the truncated Fourier series to approximate the gradient of $\boldsymbol{h}(t)$. Specifically, we present the Theorem 1, and the proof can be found in Appendix A.1.

Theorem 1. *Assume $\boldsymbol{h}(t)$ is a T-periodic and thrice differentiable function, and its second derivative $\boldsymbol{h}''(t)$ satisfies the Lipschitz condition on $[0, T]$. Then, for any $\epsilon > 0$, there exists a positive integer K_0 such that for any $K > K_0$ and for all $t \in [0, T]$, we have:*

$$\|\boldsymbol{H}'_K(t) - \boldsymbol{h}'(t)\| < \epsilon,$$

where $\boldsymbol{H}'_K(t) = \frac{2\pi}{T}\sum_{k=1}^{K} k[\boldsymbol{a}_k \sin(2\pi kt/T) + \boldsymbol{b}_k \cos(2\pi kt/T)]$. *This implies that* $\boldsymbol{H}'_K(t)$ *converges uniformly to* $\boldsymbol{h}'(t)$ *on* $[0, T]$.

In practice, we can collect a discrete-time observational time series $\boldsymbol{S} = \{\boldsymbol{s}_0, \boldsymbol{s}_1, \cdots, \boldsymbol{s}_{N-1}\}$ with a regular sampling rate at times $\{t_0, t_1, \cdots, t_{N-1}\}$. To estimate the Fourier coefficients $\tilde{\boldsymbol{S}} = \{\tilde{\boldsymbol{s}}_0, \tilde{\boldsymbol{s}}_1, \cdots, \tilde{\boldsymbol{s}}_K\}$ from the observational data $\boldsymbol{s}$, we can utilize the discrete Fourier transform (DFT) $\mathscr{F}$ and its inverse DFT (IDFT) $\mathscr{F}^{-1}$ for all $k \in \{0, 1, \cdots, K\}$ and $n \in \{0, 1, \cdots, N-1\}$, given by:

$$\begin{aligned} \tilde{\boldsymbol{s}}_k &= \mathscr{F}(\boldsymbol{S}, k) = \sum_{n=0}^{N-1} \boldsymbol{s}_n \mathrm{e}^{-i\frac{2\pi n}{N}k}, \\ \boldsymbol{s}_n &= \mathscr{F}^{-1}(\tilde{\boldsymbol{S}}, n) = \frac{1}{N}\sum_{k=0}^{K} \tilde{\boldsymbol{s}}_k \mathrm{e}^{i\frac{2\pi k}{N}n}. \end{aligned} \tag{2}$$

According to the Nyquist sampling theorem, when $N > 2K$, we can estimate the temporal derivative of the state at time t_n, which can be expressed as:

$$\hat{\boldsymbol{s}}'(t_n) = \frac{1}{N}\sum_{k=0}^{K} i\frac{2\pi k}{T}\tilde{\boldsymbol{s}}_k \mathrm{e}^{i\frac{2\pi k}{N}n} \approx \boldsymbol{s}'(t_n), \tag{3}$$

where $n \in \{0, 1, \cdots, N-1\}$.

The approximation of spatial gradients of PDEs. When considering systems described by PDEs, the state of the system evolves over both time and space, with the underlying dynamics often influenced by spatial gradients. In such cases, the ODE in Eq. (1) can be extended to:

$$\begin{aligned} \partial_t \boldsymbol{s} = \boldsymbol{f}(\boldsymbol{s}, \boldsymbol{u}, \partial_x \boldsymbol{s}, \partial_y \boldsymbol{s}, \partial_{xx}\boldsymbol{s}, \cdots),\ \boldsymbol{x} = \{x, y, \cdots\} \in \Omega, \\ \boldsymbol{s}(\boldsymbol{x}, 0) = s_0, \qquad \boldsymbol{x} \in \Omega, \end{aligned}$$

where Ω represents the spatial domain $\boldsymbol{x}$, and the system exhibits periodic boundary conditions. To accurately model the underlying dynamics $\boldsymbol{f}$, it is necessary to estimate the spatial gradients in advance. Let's consider a PDE with two spatial dimensions, denoted by x and y. The system's state at time t is sampled at the values $\boldsymbol{s}_{j_x j_y}(t) = \boldsymbol{s}(x_{j_x}, y_{j_y}, t)$, where $j_x \in \{0, 1, \ldots, N_x - 1\}$ and $j_y \in \{0, 1, \ldots, N_y - 1\}$. This leads to the following expressions:

$$\begin{aligned} \tilde{\boldsymbol{s}}_{k_x k_y} &= \sum_{j_x=0}^{N_x-1}\sum_{j_y=0}^{N_y-1} \boldsymbol{s}_{j_x j_y} \mathrm{e}^{-i2\pi\left(\frac{j_x k_x}{N_x} + \frac{j_y k_y}{N_y}\right)}, \\ \boldsymbol{s}_{j_x j_y} &= \frac{1}{N_x N_y}\sum_{k_x=0}^{K_x}\sum_{k_y=0}^{K_y} \tilde{\boldsymbol{s}}_{k_x k_y} \mathrm{e}^{i2\pi\left(\frac{k_x j_x}{N_x} + \frac{k_y j_y}{N_y}\right)}, \end{aligned}$$

where K_x and K_y represent the cutoff frequencies in the dimensions of x and y, respectively.

Denote L_x and L_y as the periodic lengths in the dimensions of x and y, respectively. And let $\boldsymbol{e}_M = (0, 1, \cdots, M-1)^\top$, $\boldsymbol{k_x} = \frac{2\pi}{L_x}(\boldsymbol{e}_{L_x}, \cdots, \boldsymbol{e}_{L_x})$, and $\boldsymbol{k_y} = \frac{2\pi}{L_y}(\boldsymbol{e}_{L_y}, \cdots, \boldsymbol{e}_{L_y})^\top$. Similar to estimating temporal gradients, we can estimate the spatial gradients for all $p, q \in \{0, 1, \cdots\}$ using the following equation:

$$\frac{\partial^{p+q}\boldsymbol{S}_t}{\partial^p x \partial^q y} = \mathscr{F}_{xy}^{-1}\left[(i\boldsymbol{k_x})^p (i\boldsymbol{k_y})^q \mathscr{F}_{xy}(\boldsymbol{S}_t)\right], \tag{4}$$

where $\boldsymbol{S}_t$ is the sampling matrix of the system at time t, with elements $\boldsymbol{S}_t[j_x, j_y] = \boldsymbol{s}_{j_x j_y}(t)$. $\mathscr{F}_{xy}$ and $\mathscr{F}_{xy}^{-1}$ represent the 2-D DFT and IDFT in space, respectively.

The framework of FNODEs. Unlike classical NODEs, we introduce the FNODEs framework, which utilizes spatiotemporal gradients obtained from Fourier analysis for more precise dynamical learning. As depicted in Figure 1, we start by estimating the gradients in the temporal and spatial directions using Eqs. (3) and (4) based on the observational time series $\boldsymbol{S}$ and the corresponding control signal $\boldsymbol{U} = \{\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_{N-1}\}$. Next, we construct a neural network $\boldsymbol{F_\theta}$ to learn the underlying dynamics of the system. Specifically, for any time t, this neural network takes the system state $\boldsymbol{s}$, control signal $\boldsymbol{u}$, and spatial gradients $\{\partial_x \boldsymbol{s}, \partial_y \boldsymbol{s}, \cdots\}$ (only for PDEs) as inputs, and outputs the prediction of the temporal gradient at time t. During this process, we update the parameters $\boldsymbol{\theta}$ using the following gradient flow matching loss function:

$$\mathcal{L} = \frac{1}{N}\sum_{n=0}^{N-1}\left\|\hat{\boldsymbol{h}}'(t_n) - \boldsymbol{F_\theta}[\boldsymbol{s}(t_n), \boldsymbol{u}(t_n), \partial_x \boldsymbol{s}(t_n), \cdots]\right\|. \tag{5}$$

Notably, in the training process, there is no need to utilize the ODE solvers, resulting in a simulation-free training framework.

Data augmentation. In addition, we incorporate a data augmentation strategy to facilitate the modeling of complex systems. This strategy involves simulating the trained NODEs to obtain the system's states at arbitrary times, thereby increasing the sampling frequency. By generating a larger sample size, we can better satisfy the requirements of the Nyquist sampling theorem, leading to more accurate estimations of the spatiotemporal gradients. The newly generated data is then fed into our model for further training. This two-step process, consisting of data augmentation and training, forms a positive feedback loop that enhances the accuracy and robustness of our approach.

It is important to note that the augmentation strategy does not participate in the computational graph, resulting in a low computational cost limited to the forward simulation. In contrast, the classical NODEs framework includes simulation in the training process, which incurs a high computational cost due to solving extremely high-dimensional ODEs in the backward pass. The entire framework is illustrated in Fig. 1, and we provide a detailed execution process in Algorithm 1 in Appendix A.2.

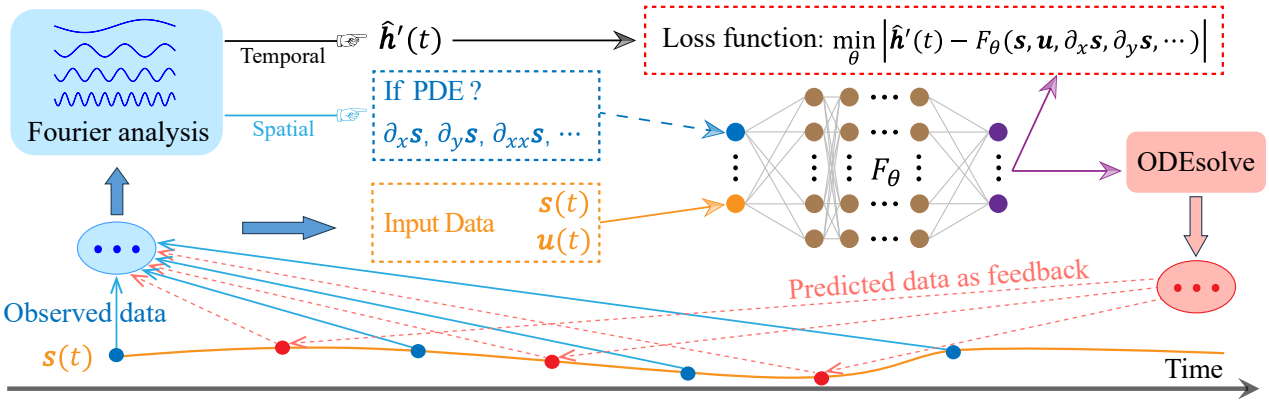

Figure 1. Illustration of the FNODEs framework.

4. Illustrative Experiments

In this section, we present a thorough analysis of our framework based on experiments conducted using a computational setup with 64GB RAM and an NVIDIA Tesla V100 GPU equipped with 16GB memory. Specifically, we generate a set of control functions u using Gaussian random fields (GRFs) with a radial-basis function kernel, as defined by

$$u \sim c \cdot \mathcal{G}\left\{\mu, \exp\left[||x_1 - x_2||^2/(2l^2)\right]\right\},$$

where μ represents the mean value, l is the length scale that governs the smoothness of the sampling function, and c is the scaling factor of the output. To validate our framework, we consider experimental data $\boldsymbol{s}(\boldsymbol{x}, t, \boldsymbol{u}(\boldsymbol{x}, t))$ derived from multiple dynamical systems, and compare our results with several state-of-the-art baseline methods, including NODEs (Chen et al., 2018), Deep Operator Network (DeepONet) (Lu et al., 2019), Fourier Neural Operator (FNO) (Li et al., 2020), PDE-Net (Long et al., 2018), Physics-Informed Neural Networks (PINN) (Raissi et al., 2019), Deep Auto-Regressive (DeepAR) (Flunkert et al., 2017), and the message-passing PDE solvers (MP-PDE) (Brandstetter et al., 2022). Detailed descriptions of these baseline methods can be found in Appendix D.

4.1. Parametric ODEs

We initially consider a 2-D parametric ODE system, which is described by

$$\frac{\mathrm{d}\boldsymbol{s}}{\mathrm{d}t} = \begin{bmatrix} -0.1 & 2.0 \\ -2.0 & -0.1 \end{bmatrix} \boldsymbol{s}^3 + \boldsymbol{u}(t), \quad \boldsymbol{s}(0) = \boldsymbol{s}_0 \tag{6}$$

where $\boldsymbol{u}(t) = [u_1(t), u_2(t)]^\top$ is a vector-valued function derived from Gaussian Random Fields (GRF). We generate a dataset comprising 100 training samples, 30 validation samples, and 50 testing samples, with $\mu = 0$, $c = 100$, $l = 0.1$, and random initial values $\boldsymbol{s}_0$.

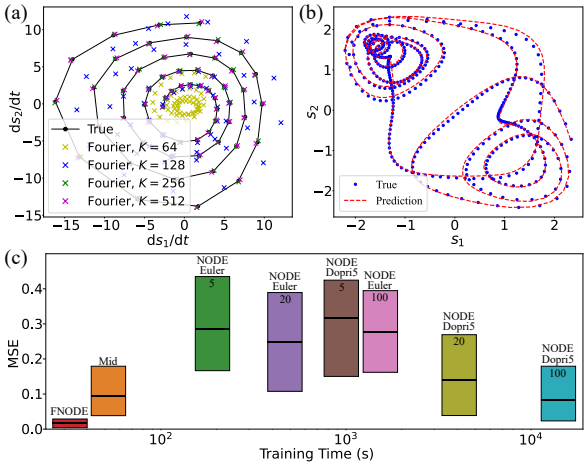

Figure 2. The experimental results of system (6) using FNODEs. (a) Estimation of the derivatives based on the observational data using Fourier analysis. (b) The test prediction using the trained model. (c) Comparison of the FNODEs and baseline methods in terms of training time and prediction error.

Here, we randomly select a segment of data $\boldsymbol{S}_g$, and each temporal gradient of the state at the time t in $\boldsymbol{S}_g$ is estimated directly using Eq. (3). The estimation of temporal gradients for different truncation frequencies K is illustrated in Figure 2(a). It is evident that Eq. (3) is not effective in approximating the true gradients when K is small. However, when K exceeds a certain threshold, Fourier analysis can effectively approximate the true gradients. In our experiment, with $N = 500$ sampling points, the maximum value of K for recoverable signals, as per the Nyquist sampling theorem, is 250. Therefore, setting K to 512 does not yield better performance compared to setting it to 256. Moreover, our framework exhibits robustness to approximation errors in gradient estimation from a statistical perspective due to the inclusion of multiple data points under different control signals in the training set.

After training our model, we assess its performance on the

Table 1. Comparison of the prediction MSE and the training time of baseline methods under different training set sizes and noise intensities. Here, we consider zero-mean Gaussian noise with $N_{\text{tr}} = 1000$, and the standard deviation is σ_{sd} times the mean absolute value of the data.

Models	Training set sizes N_{tr}				Noise intensify σ_{sd}				Training time (s)
	10	20	100	1000	0.1%	1%	5%	10%	
Mid	0.129	0.102	0.075	0.072	0.103	0.115	0.134	0.285	120.1
DeepAR	3.467	2.318	0.633	0.801	0.783	0.884	1.386	2.297	101.8
NODE (Euler)	0.211	0.191	0.157	0.132	0.133	0.193	0.220	0.280	631.8
NODE (Dopri5)	0.156	0.096	0.087	0.062	0.111	0.172	0.188	0.264	9632.2
FNODE	**0.049**	**0.030**	**0.018**	**0.012**	**0.016**	**0.013**	**0.083**	**0.184**	**62.4**

test data. As depicted in Figure 2(b), our framework consistently achieves high prediction accuracy for long-term forecasting. To further demonstrate its effectiveness, we introduce a new baseline method called the "Mid" method, which estimates gradients using the classical central difference numerical method. Additionally, we conduct experiments using the NODEs with two numerical ODE solvers, namely "Euler" and "Dopri". Figure 2(c) displays the training time and the prediction mean squared error (MSE) of the different baselines on the test data. It is evident that the FNODEs require the shortest training time and simultaneously achieve the lowest prediction error.

We note that the classical NODE method tends to get trapped in local optima in this particular example, resulting in longer training times. To further verify the robustness of our proposed framework, we conduct experiments with varying training set sizes and noise intensities. Table 1 demonstrates that the FNODEs outperform the baselines in our experiments, showcasing stronger robustness and superior prediction capabilities.

4.2. Parametric PDEs

To further assess the effectiveness of our framework, we first conduct experiments on various common parametric PDEs, including the Korteweg-de Vries (KDV) system,

$$\partial_t s = \partial_{xxx} s + u(x,t)\partial_x(s^2/2),\ x \in [0, 16\pi],\ t \in [0, 20],$$

the diffusion-reaction (DR) system,

$$\partial_t s = 0.01\partial_{xx} s + 0.01 s^2 + u(x,t),\ x \in [0, 1],\ t \in [0, 1],$$

and the Kuramoto-Sivashinsky (KS) system,

$$\partial_t s = -\partial_x(s^2/2) - \partial_{xx} s - u(x,t)\partial_{xxxx} s,$$

where $x \in [0, 32\pi],\ t \in [0, 20]$. Then we also consider a 2-D Navier-Stokes (NS) system, given by

$$\partial_t s = \partial_x \gamma \partial_y s - \partial_y \gamma \partial_x s + 0.001\Delta s + u(x,y,t),\ \ \Delta\gamma = -s,$$

where $(x, y) \in [0, 2]^2,\ t \in [0, 10]$.

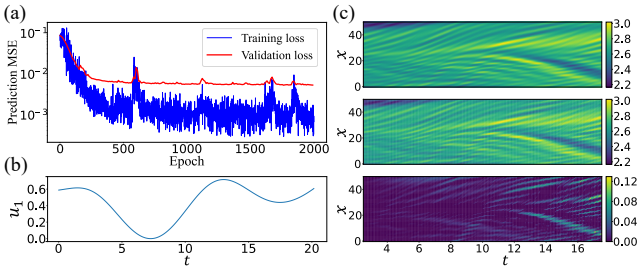

Figure 3. The experimental results of the KDV system using FNODEs Method. (a) The variation of the training and validation loss during the training process. (b) Plot of $u_1(t)$ from a test data. (c) These three subplots from top to bottom represent the ground truth, predicted results, and prediction errors of test data.

Korteweg-de Vries system. We begin by considering the KDV system, which is commonly used to describe the evolution of water waves. In this equation, $s(x,t)$ represents the wave displacement, x denotes the spatial coordinate and t represents time. The control parameter is denoted as $u(x,t)$. In this example, we consider $u(x,t) = \sin(x/8) + u_1(t)$, where $u_1(t)$ is a 1-D function sampled from GRFs. To validate the effectiveness of our framework, we fix the initial value s_0 and generate 100 training data, 30 validation data, and 30 test data. During the training process, we estimate the spatial partial derivatives $\mathscr{F}^{-1}\{i\boldsymbol{k_x}, (i\boldsymbol{k_x})^2, (i\boldsymbol{k_x})^3, (i\boldsymbol{k_x})^4\}$ using Eq. (4) as additional inputs to the neural network $\boldsymbol{F_\theta}$. Then, we employ the gradient flow matching loss Eq. (5) to train our model.

During the training process, both the training loss and validation loss rapidly decrease to lower levels (see Fig. 3(a)). The entire training process is completed in just 28 seconds for 2000 epochs, indicating the efficiency of our framework. In addition, our framework exhibits outstanding predictive performance on the test data with an average prediction error of only 0.04. Figures 3(b)-(c) depict the prediction results for one sample of the test data set, clearly demonstrating the sustained high prediction accuracy over an extended period, successfully revealing the underlying dynamics of the KDV system. For the 1-D DR and KS, we provide the corresponding results in Appendix C in a similar manner.

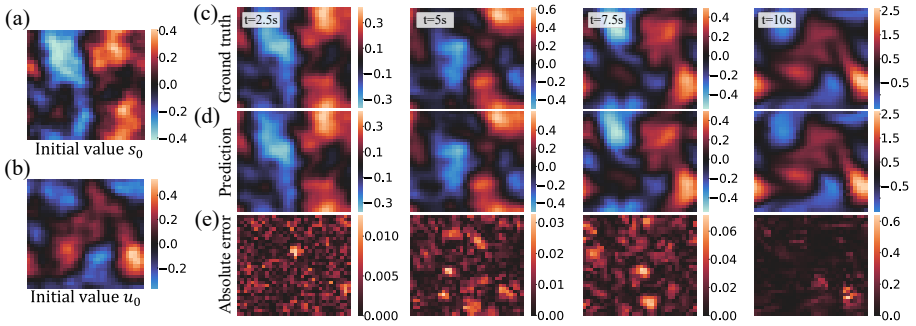

Figure 4. The experimental results of the NS system using FNODEs Method. (a) The initial state s_0 of a certain test data. (b) The initial state u_0 of the parameter $u(x, y, 0)$. (c)-(e) These subfigures depict the ground truth, prediction results, and prediction errors at different time instances. From left to right, the time instances t correspond to 2.5s, 5s, 7.5s, and 10s.

Table 2. Prediction MSE under different methods, systems, and experimental conditions.

Models	$N_{\text{tr}} = 10$, $\sigma_{\text{sd}} = 0$				$N_{\text{tr}} = 1000$, $\sigma_{\text{sd}} = 0$				$N_{\text{tr}} = 1000$, $\sigma_{\text{sd}} = 5\%$			
	KDV	DR	KS	NS	KDV	DR	KS	NS	KDV	DR	KS	NS
NODEs	1.93	1.68	2.56	4.21	0.45	0.58	1.69	3.61	1.02	1.46	2.34	5.19
DeepONet	0.83	2.75	3.02	38.3	0.09	0.17	0.63	7.30	0.74	0.87	1.20	14.7
FNO	1.88	1.56	2.21	23.9	0.21	0.13	0.92	5.3	0.89	1.58	2.13	12.6
PDE-NET	1.64	1.38	2.32	2.09	0.11	0.27	0.78	0.92	1.34	0.97	1.73	2.30
PINN	**0.17**	**0.19**	**0.24**	**0.31**	**0.17**	**0.19**	**0.24**	**0.31**	**0.17**	**0.19**	**0.24**	**0.31**
MP-PDE	2.39	2.74	4.23	6.62	0.44	0.49	1.33	2.07	1.22	1.11	1.93	2.59
FNODEs	**0.23**	**0.16**	**0.28**	**0.29**	**0.023**	**0.016**	**0.11**	**0.083**	**0.14**	**0.12**	**0.46**	**0.11**

Navier-Stokes system. Next, we turn our attention to the more complex 2-D NS system in the vorticity form, which involves two spatial dimensions, namely x and y. In this case, we consider $u(x, y, t) = u_0(x, y) + u_1(t)$, where u_0 and u_1 are 1-D functions sampled from GRFs. Similar to the process in the KDV system, we incorporate the estimated spatial partial derivatives as inputs into the neural network using Eq. (4).

In addition to the conventional terms

$$\mathscr{F}^{-1}\{(i\boldsymbol{k_x}), (i\boldsymbol{k_y}), (i\boldsymbol{k_x})^2, (i\boldsymbol{k_y})^2, (-\boldsymbol{k_x}\boldsymbol{k_y}), \cdots\}$$

we introduce two additional prior terms $\mathscr{F}^{-1}[i\boldsymbol{k_x}/(\boldsymbol{k_x^2} + \boldsymbol{k_y^2})]$ and $\mathscr{F}^{-1}[i\boldsymbol{k_y}/(\boldsymbol{k_x^2} + \boldsymbol{k_y^2})]$ to better capture the intrinsic variable γ in the equation. Consequently, our framework successfully learns the dynamics of the NS system, achieving an average prediction error of only 0.03. We present the predicted results of the test data in Fig. 4, which clearly demonstrate the excellent predictive performance of our framework within the specified time range. Moreover, the FNODEs exhibit fast training speed in a runtime of 283 seconds for 5000 epochs.

Evaluation against baseline methods. To substantiate the efficacy of our proposed framework, we compare it with the state-of-the-art techniques, considering factors such as the size of the training set, robustness to noise, and training time. Specifically, we conduct experiments on four distinct PDE systems under three varying experimental configurations, and the results are shown in Table 3. It is unequivocally clear that our framework is capable of discerning the underlying dynamics in the shortest time, particularly surpassing the standard NODEs method by more than 10 times. For an in-depth understanding of the baseline methods, please refer to Appendix D.

Table 3. Comparing the training time of different methods in multiple experiments. Here, "Sys" denotes system, "P-N" refers to the PDE-NET method and "M-P" represents the MP-PDE method.

Sys	NODEs	DeepONet	FNO	P-N	PINN	M-P	FNODEs
KDV	3938	751	271	398	1540	560	**130**
DR	2910	652	262	311	1532	367	**65**
KS	5120	641	241	492	1660	589	**74**
NS	21747	8484	549	1323	2430	734	**283**

Furthermore, Table 2 illustrates the MSE of the test data under diverse experimental configurations. It is observed that our proposed framework consistently yields lower prediction errors in comparison to the baseline methods. Here, it should be noted that our framework may require a longer prediction time compared to neural operator methods such as DeepONet and FNO, it exhibits superior performance in terms of the generalization, especially for NS systems with random initial values. Additionally, our framework

surpasses the performance of the PDE-NET, thereby further validating the effectiveness of Fourier analysis-based estimation of spatiotemporal gradients over traditional numerical gradient estimation methods. As for the MP-PDE method, as a black-box method in deep learning, it did not demonstrate satisfactory predictive performance in our experiments with a limited training set. In addition, the PINN method demonstrates performance similar to our approach, exhibiting robustness against fluctuations in training set size and noise levels. This is attributable to the necessity of employing underlying dynamical equations for training. Therefore, in data-driven modelling tasks (PINN is not applicable), our method indeed learned the underlying dynamics, demonstrating significant advantages in predictive capability.

4.3. Additional Validation and Discussion of Our Framework

4.3.1. Handling Sparse and Irregularly Sampled Time Series.

Our framework can be extended to the case of irregularly sampled time series by replacing Eqs. (2) and (3) with the nonuniform discrete Fourier transform (NDFT). The computation of NDFT can be expedited using the nonuniform fast Fourier transform (NFFT) (Dutt & Rokhlin, 1993; Barnett et al., 2018). Moreover, in scenarios where the number of sampling data points is limited, the restorable signal frequency $N/2$, as per the Nyquist sampling theorem, may be less than the cutoff frequency K of the true signal. Under such circumstances, Fourier analysis may fail to accurately estimate the temporal gradients of the system. However, our data augmentation training strategy can circumvent this obstacle by integrating the predicted results of the FNODEs into the original dataset, thereby significantly enhancing the data utilization.

4.3.2. Resolution-invariant property of the FNODEs.

To investigate the relationship between the truncated frequency K and the number of sampling points N, we conduct experiments on the system (6) by generating data with varying sparsity levels, while maintaining other hyperparameters consistent with the previous section. As depicted in Fig. 5(a), we employ Fourier analysis to estimate the temporal gradients at different sparsity levels. The optimal truncated frequency for each dataset is almost near $N/2$, which underscores the non-negligible role of Fourier basis with $K < 256$ in system modeling. Therefore, a larger number of sampling points N allows for a higher usable cutoff frequency K, facilitating the capture of the true signal gradients.

Then, we initially consider 100 training time series with only 200 sampling points with $K = 100$. Under this setting, the prediction of the trained model on test data is displayed in Fig. 5(c). Subsequently, we use the trained model to predict 200 additional data points to augment the training data and retrain our model with $K = 200$. The prediction result in this case is depicted in Fig. 5(d). In the same manner, we further predict 400 points and set $K = 400$ to train our model. The corresponding predicted result is shown in Fig. 5(e). Moreover, Figure 5(b) presents the boxplots of prediction errors on all test data under previous settings. It is clear that the data augmentation training strategy can further enhance the modeling performance without measuring additional data samples, which may be prohibited in the real-world experiments.

Indeed, our framework can be directly employed for resolution-invariant modeling of PDE systems. In the temporal dimension, the employment of ODEsolve in the trained model can predict the system state at an arbitrary time point. For the spatial dimension, our model can be trained on data with lower sampling frequencies but make predictions on data with higher sampling frequencies. To verify this property, we train our model on training data of the DR system with N_x equal to 32, 64, and 128, respectively, and make predictions on the test set with $N_x = 128$. The corresponding prediction errors are shown in Fig. 5(f). Figure 5(g), as a case study, is the prediction with the number of spatial samples $N_x = 128$ based on the trained model using the lower sampling frequency data with $N_x = 32$ as shown in Fig. 5(h). It is evident that our framework can achieve effective predictions of PDEs with zero-shot super-resolution.

4.3.3. Experimental results on a real-world system

To explore the potential applicability of the method presented in this paper in real-world systems, we conducted preliminary experiments in the time series data of polar motion (the data can be accessed via `https://www.iers.org/IERS/EN/DataProducts/EarthOrientationData/eop.html`). The polar motion components as the crucial Earth Orientation Parameters (EOPs), denoted usually by x_p and y_p, describe the position of the Earth's instantaneous rotation axis with respect to the Earth's surface. In the field of geodesy, it is necessary to have accurate predictions of these parameters.

To enhance the accuracy of the dynamics modeling and prediction, we incorporate multiple distinct sources of physical information as features (Shahvandi et al., 2022). Specifically, we utilize $dUT1 = UT1 - UTC$ and Length of Day (LOD) as the initial two features. Additionally, we also incorporate four Effective Angular Momentum (EAM) functions as features, namely: (a) Atmospheric Angular Momentum (AAM); (b) Hydrological Angular Momentum (HAM); (c) Oceanic Angular Momentum (OAM); and (d) Sea-

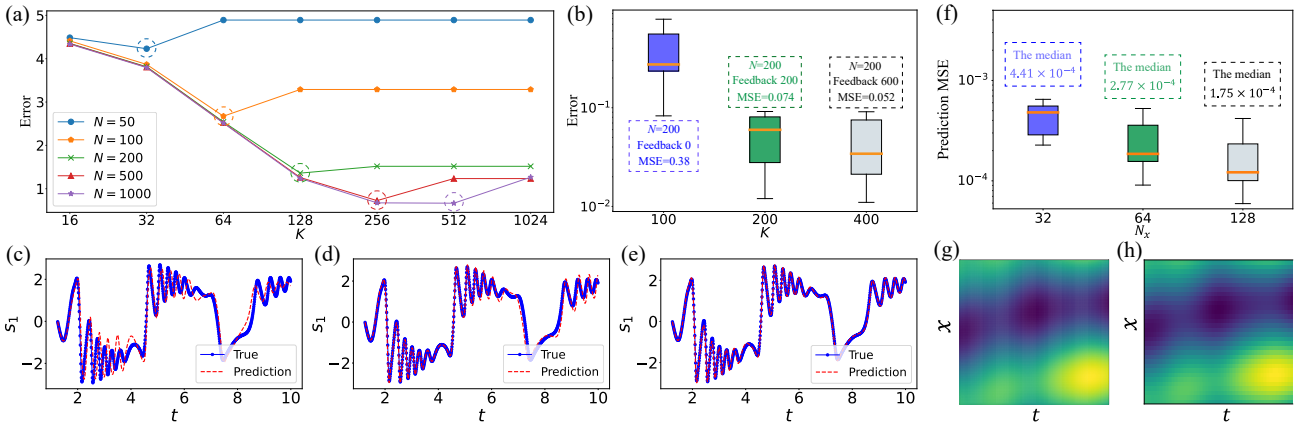

Figure 5. Experimental results of the data augmentation and the resolution-invariant predictions using the FNODEs. In system (6), (a) Prediction errors of time gradients under different sampling rates N and cutoff frequencies K. (b) A boxplot depicting the testing errors at different stages under the utilization of feedback data, with the median of prediction errors represented by a yellow horizontal line. (c)-(e) The prediction results of test data under different feedback settings (0, 200, and 600 feedback points respectively). In DR system, (f) The prediction MSE of temporal gradient under different N_x in training data. (g) A training sample with a low resolution. (h) A predicted sample with a high resolution.

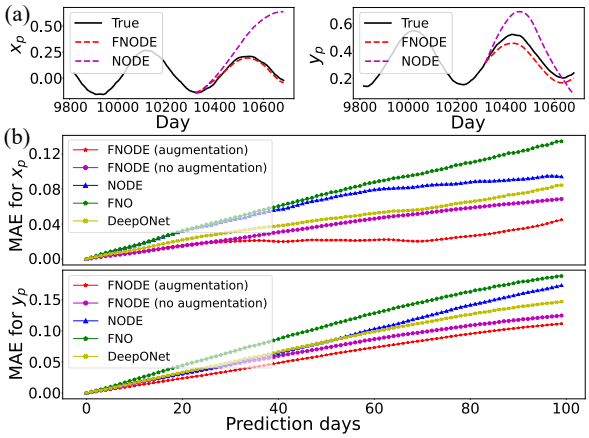

Figure 6. Experimental results on the polar motion dataset. (a) The prediction performance for a data segment using FNODE and NODE methods. (b) The prediction results for the polar motion data using different methods. Here, MAE refers to mean absolute error.

Level Angular Momentum (SLAM), and these data can be accessed via `http://rz-vm115.gfz-potsdam.de:8080/repository`.

We partition the data as follows: 70% for training, 10% for validation, and the remaining 20% for testing. As shown in Fig. 6(a), our approach not only outperforms the traditional NODE method in prediction accuracy, but also reduces the training time by over an order of magnitude. Furthermore, we conducted a prediction for the subsequent 100 days from 50 randomly selected starting points under various methods, with the results depicted in Fig. 6(b). It is discernible that our method outperforms the baselines in both short-term and long-term predictions. Additionally, we also conduct a ablation experiment on data augmentation strategies (see FNODE (augmentation) and FNODE (no augmentation) in Fig. 6(b) for details). It is apparent from the results that incorporating feedback mechanisms during the training process can further enhance the predictive performance of the model.

5. Concluding Remarks

In this article, we propose the FNODEs framework, a novel approach for efficient and robust dynamical system modeling. Our framework addresses the challenges faced by standard NODEs, including high computational cost and susceptibility to local optima. The proposed framework combines Fourier analysis and NODEs to transform the modeling problem of complex dynamics into a gradient flow matching task.

We validate our framework on multiple ODE and PDE systems, and experimental results demonstrate that our framework exhibits significantly faster training speed, surpassing the standard NODE by more than tenfold. The FNODEs outperform NODEs and other state-of-the-art methods in various prediction tasks, particularly demonstrating strong robustness in noisy data. Furthermore, we validate, both theoretically and experimentally, that the data augmentation training strategy extremely enhances the maximum truncated frequency of Fourier analysis, facilitating the spatiotemporal gradient estimations and system modeling.

However, our method has certain limitations. For instance, estimating the gradient flow within an acceptable truncated frequency using Fourier analysis becomes challenging when

dealing with more complex systems. One potential direction for resolution is to extend our framework by combining Koopman and uncertainty quantification theories. Additionally, our method exhibits inefficiency similar to NODEs when predicting high-dimensional systems, necessitating further optimization by integrating the idea of neural operators.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgments

Q. Zhu is supported by the China Postdoctoral Science Foundation (No. 2022M720817), by the Shanghai Postdoctoral Excellence Program (No. 2021091), and by the STCSM (Nos. 21511100200, 22ZR1407300, 22dz1200502, and 23YF1402500). W. Lin is supported by the NSFC (Grant No. 11925103), by the STCSM (Grants No. 22JC1402500 and No. 22JC1401402), and by the SMEC (Grant No. 2023ZKZD04). The computational work presented in this article is supported by the CFFF platform of Fudan University.

References

Barnett, A. H., Magland, J. F., and af Klinteberg, L. A parallel non-uniform fast fourier transform library based on an "exponential of semicircle" kernel. *ArXiv*, abs/1808.06736, 2018. URL https://api.semanticscholar.org/CorpusID:52055548.

Bilovs, M., Sommer, J., Rangapuram, S. S., Januschowski, T., and Gunnemann, S. Neural flows: Efficient alternative to neural odes. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:239768892.

Brandstetter, J., Worrall, D. E., and Welling, M. Message passing neural pde solvers. *ArXiv*, abs/2202.03376, 2022.

Carroll, T. L. Testing dynamical system variables for reconstruction. *Chaos*, 28 10:103117, 2018.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Neural Information Processing Systems*, 2018.

Ding, F., Xu, L., Alsaadi, F. E., and Hayat, T. Iterative parameter identification for pseudo-linear systems with arma noise using the filtering technique. *Iet Control Theory and Applications*, 12:892–899, 2018. URL https://api.semanticscholar.org/CorpusID:125115188.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. *ArXiv*, abs/1904.01681, 2019.

Dutt, A. and Rokhlin, V. Fast fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14:1368–1393, 1993.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. M. How to train your neural ode. *arXiv preprint arXiv:2002.02798*, 2, 2020.

Flunkert, V., Salinas, D., and Gasthaus, J. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *ArXiv*, abs/1704.04110, 2017. URL https://api.semanticscholar.org/CorpusID:12199225.

Holt, S., Qian, Z., and van der Schaar, M. Neural laplace: Learning diverse classes of differential equations in the laplace domain. In *International Conference on Machine Learning*, 2022.

Kaiser, E., Kutz, J. N., and Brunton, S. L. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings. Mathematical, Physical, and Engineering Sciences*, 474, 2017. URL https://api.semanticscholar.org/CorpusID:49216558.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural controlled differential equations for irregular time series. *ArXiv*, abs/2005.08926, 2020.

Kim, S., Ji, W., Deng, S., and Rackauckas, C. Stiff neural ordinary differential equations. *Chaos*, 31 9:093122, 2021. URL https://api.semanticscholar.org/CorpusID:232404648.

Lanzieri, D., Lanusse, F., and Starck, J.-L. Hybrid physical-neural odes for fast n-body simulations. *ArXiv*, abs/2207.05509, 2022. URL https://api.semanticscholar.org/CorpusID:250451236.

Legaard, C. M., Schranz, T., Schweiger, G., Drgovna, J., Falay, B., Gomes, C., Iosifidis, A., Abkar, M., and Larsen, P. G. Constructing neural network based models for simulating dynamical systems. *ACM Computing Surveys*, 55:1 – 34, 2021. URL https://api.semanticscholar.org/CorpusID:240420173.

Li, X., Zhu, Q., Zhao, C., Qian, X., Zhang, X., Duan, X., and Lin, W. Tipping point detection using reservoir computing. *Research*, 6:0174, 2023.

Li, X., Zhu, Q., Zhao, C., Duan, X., Zhao, B., Zhang, X., Ma, H., Sun, J., and Lin, W. Higher-order granger reservoir computing: simultaneously achieving scalable complex structures inference and accurate dynamics prediction. *Nature Communications*, 15(1):2506, 2024.

Li, Z.-Y., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. M., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *ArXiv*, abs/2010.08895, 2020. URL https://api.semanticscholar.org/CorpusID:224705257.

Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. Neural sde: Stabilizing neural ode networks with stochastic noise. *ArXiv*, abs/1906.02355, 2019.

Long, Z., Lu, Y., and Dong, B. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.*, 399, 2018. URL https://api.semanticscholar.org/CorpusID:54559221.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218 – 229, 2019. URL https://api.semanticscholar.org/CorpusID:233822586.

Meiss, J. D. Differential dynamical systems. In *Mathematical modeling and computation*, 2007. URL https://api.semanticscholar.org/CorpusID:117015674.

Pathak, J., Hunt, B. R., Girvan, M., Lu, Z., and Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120 2:024102, 2018. URL https://api.semanticscholar.org/CorpusID:206306411.

Perko, L. M. Differential equations and dynamical systems. 1991. URL https://api.semanticscholar.org/CorpusID:117371654.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019. URL https://api.semanticscholar.org/CorpusID:57379996.

Sanhedrai, H., Gao, J., Bashan, A., Schwartz, M., Havlin, S., and Barzel, B. Reviving a failed network through microscopic interventions. *Nature Physics*, 18:338–349, 2020.

Shahvandi, M. K., Schartner, M., and Soja, B. Neural ode differential learning and its application in polar motion prediction. *Journal of Geophysical Research: Solid Earth*, 127, 2022. URL https://api.semanticscholar.org/CorpusID:253393259.

Zappala, E., de Oliveira Fonseca, A. H., Moberly, A. H., Higley, M. J., Abdallah, C. G., Cardin, J. A., and van Dijk, D. Neural integro-differential equations. In *AAAI Conference on Artificial Intelligence*, 2022.

Zhang, Q. and Zhou, Y. Recent advances in non-gaussian stochastic systems control theory and its applications. *International Journal of Network Dynamics and Intelligence*, pp. 111–119, 2022.

Zhu, Q., Guo, Y., and Lin, W. Neural delay differential equations. *ArXiv*, abs/2102.10801, 2021.

Zhu, Q., Li, X., and Lin, W. Leveraging neural differential equations and adaptive delayed feedback to detect unstable periodic orbits based on irregularly sampled time series. *Chaos*, 33 3:031101, 2023.

Zhuang, J., Dvornek, N. C., Li, X., Tatikonda, S. C., Papademetris, X., and Duncan, J. S. Adaptive checkpoint adjoint method for gradient estimation in neural ode. *Proceedings of machine learning research*, 119:11639–11649, 2020. URL https://api.semanticscholar.org/CorpusID:219303652.

Appendix

A. Theorems and Algorithms

A.1. Proof of Theorem 1

We aim to prove that the derivative of the truncated Fourier series converges uniformly to $\boldsymbol{h}'(t)$. Let's denote $\boldsymbol{H}_K(t)$ as the partial sum of the Fourier series of $\boldsymbol{h}(t)$:

$$\boldsymbol{H}_K(t) = \boldsymbol{a}_0 + \sum_{k=1}^{K}[\boldsymbol{a}_k \cos(\frac{2\pi}{T}kt) + \boldsymbol{b}_k \sin(\frac{2\pi}{T}kt)],\ \ \boldsymbol{h}'(t) = \frac{2\pi}{T}\sum_{k=1}^{\infty}[k\boldsymbol{b}_k \cos(\frac{2\pi}{T}kt) - k\boldsymbol{a}_k \sin(\frac{2\pi}{T}kt)]. \tag{S1}$$

We consider the derivative of the partial sum of the Fourier series of $\boldsymbol{h}(t)$, denoted as $\boldsymbol{H}'_K(t)$:

$$\boldsymbol{H}'_K(t) = \frac{2\pi}{T}\sum_{k=1}^{K}[k\boldsymbol{b}_k \cos(\frac{2\pi}{T}kt) - k\boldsymbol{a}_k \sin(\frac{2\pi}{T}kt)].$$

We define $\boldsymbol{G}_K(t) = \boldsymbol{h}'(t) - \boldsymbol{H}'_K(t)$, so we have:

$$\boldsymbol{G}_K(t) = \frac{2\pi}{T}\sum_{k=K+1}^{\infty}[k\boldsymbol{b}_k \cos(\frac{2\pi}{T}kt) - k\boldsymbol{a}_k \sin(\frac{2\pi}{T}kt)].$$

Since $\boldsymbol{h}''(t)$ satisfies the Lipschitz condition, there exists a Lipschitz constant M such that:

$$\|\boldsymbol{h}''(t_1) - \boldsymbol{h}''(t_2)\| \leq M_0|t_1 - t_2|, \quad \forall t_1, t_2 \in [-\pi, \pi],$$

where M_0 is a positive real number. Given the existence of $\boldsymbol{h}$'s third derivative, we have:

$$|\boldsymbol{h}'''(t)| = \lim_{\boldsymbol{h}\to 0}\left\|\frac{\boldsymbol{h}''(t+h) - \boldsymbol{h}''(t)}{h}\right\| \leq M, \quad t \in [-\pi, \pi).$$

Consequently, M_0 also serves as the upper bound of the third-order derivative of $\boldsymbol{h}$. Now, we can find the upper bounds for $|\boldsymbol{a}_k|$ and $|\boldsymbol{b}_k|$. We have:

$$\boldsymbol{a}_k = \frac{2}{T}\int_0^T \boldsymbol{h}(t)\cos(\frac{2\pi}{T}kt)dt, \quad \boldsymbol{b}_k = \frac{2}{T}\int_0^T \boldsymbol{h}(t)\sin(\frac{2\pi}{T}kt)dt.$$

Integrating by parts three times, we get:

$$\boldsymbol{a}_k = \frac{1}{k^3} \times \frac{T^2}{4\pi^3}\int_{-\pi}^{\pi} \boldsymbol{h}'''(t)\cos(\frac{2\pi}{T}kt)dt, \quad \boldsymbol{b}_k = -\frac{1}{k^3} \times \frac{T^2}{4\pi^3}\int_{-\pi}^{\pi} \boldsymbol{h}'''(t)\sin(\frac{2\pi}{T}kt)dt.$$

Using the triangle inequality, we have:

$$|\boldsymbol{a}_k| \leq \frac{M_1}{k^3}\int_0^T |\boldsymbol{h}'''(t)|dt, \quad |\boldsymbol{b}_k| \leq \frac{M_1}{k^3}\int_0^T |\boldsymbol{h}'''(t)|dt,$$

where $M_1 = T^2/(4\pi^3) > 0$. Then we have:

$$|\boldsymbol{a}_k|, |\boldsymbol{b}_k| \leq \frac{M_0 M_1 T}{k^3}.$$

Let $M = M_0 M_1 T > 0$, we get:

$$|\boldsymbol{G}_N(t)| \leq \sum_{k=K+1}^{\infty} \frac{M}{k^3} |k\cos(kt) - k\sin(kt)| \leq \sum_{k=K+1}^{\infty} \frac{M}{k^2}.$$

The right-hand side is a p-series, where $p = 2 > 1$, which is a convergent series. Therefore, as $K \to \infty$, we have $|\boldsymbol{G}_K(t)| \to 0$. This means that $\boldsymbol{H}'_K(t)$ converges uniformly to $\boldsymbol{h}'(t)$. This completes the proof.

A.2. Execution process of the FNODEs framework

To provide a clearer description of our method's execution process, we present a concise procedure as Algorithm 1.

Algorithm 1 Execution process of the FNODEs framework

Input: The observed data $\boldsymbol{s}(t_n)$, $n \in \{0, 1, \cdots, N-1\}$; Control parameter $\boldsymbol{u}(t)$.
Results: The trained agent model F_θ.
Step1: Set an appropriate truncation frequency K, neural network F_θ;
Step2: Estimate the temporal gradient $\hat{\boldsymbol{h}}'(t_n)$ using Eqs. (2) and (3) of the main text, and the spatial gradient (if it is a PDE) using Eqs. (5) and (6) of the main text;
Step3: Compute the loss function using Eq. (7) of the main text and update the parameters θ;
Step4: If the prediction results of F_θ have a validation loss below a certain threshold, the algorithm terminates; otherwise, proceed to the next step;
Step5: Predict $\boldsymbol{s}((t_n + t_{n+1})/2)$ using F_θ, where $n \in \{0, 1, \cdots, N-2\}$;
Step6: Let $N \leftarrow 2N - 1$ and merge the observed data and predicted data in chronological order, still denoted as $\boldsymbol{s}(t_n)$, $n \in \{0, 1, \cdots, N-1\}$. Return to Step 2.

B. Experimental Hyperparameters

In this section, we present the hyperparameter settings for all experiments conducted in the main text and appendix. First, we construct a multi-layer fully connected neural network to model the underlying dynamics $\boldsymbol{f}$, and use the neural network to predict the vector field of the dynamical system from a dynamical perspective. The neural network consists of four hyperparameters: the input dimension d_i, the hidden layer dimension d_h, the number of hidden layers l_h, and the output dimension d_o. Regarding the training data, it comprises three hyperparameters: the number of training samples N_{tr}, the number of time domain sampling points N, and the number of spatial domain sampling points N_x and/or N_y. For Fourier analysis, it is necessary to consider the truncation frequency K. For the training process, In Gaussian random field sampling, we consider the output scale c, mean μ, and length scale l. Therefore, unless otherwise specified, the hyperparameter settings for our experiments are shown in Table S1.

During the training process, we set the learning rate to 0.001, weight decay to 1e-5, and employed the Adam optimizer for training. In addition, the initial values for the NS and 2D-ODE experiments are randomly generated, while for the KDV, DR, and KS experiments, the initial values are given as follows:

$$\begin{aligned} s_{\text{KDV}} &= 2\cos(x/L_{\text{KDV}}) \times (1 + \sin(x/L_{\text{KDV}})), \\ s_{\text{DR}} &= \cos(2\pi x/L_{\text{DR}}), \\ s_{\text{KS}} &= 2\cos(x/L_{\text{KS}}) \times (1 + \sin(x/L_{\text{KS}})), \end{aligned}$$

where L_{KDV}, L_{DR}, and L_{KS} represent the spatial lengths of the KDV, DR, and KS systems, respectively.

Table S1. Experimental hyperparameters in different systems

Experiment	N_{tr}	N	$N_{\boldsymbol{x}}$	d_i	μ	l	c	K	l_h
2-D ODE	100	1000	/	3	0	0.1	20	300	3
KDV PDE	100	1000	64	6	0	0.2	1.0	512	3
DR PDE	100	1000	64	6	0	0.1	1.0	512	3
KS PDE	100	1000	64	6	2	0.2	1.0	512	3
NS PDE	100	1000	32×32	8	0	0.1	1.0	256	4

C. Experimental Details

In this section, we provide additional experiments and explanations to illustrate the content in the main text better.

C.1. Potential spatial gradients

To improve the modeling of PDE systems, we consider the potential spatial derivative terms estimated from equations (7) and (8) as additional inputs to the neural network. In fact, these partial derivatives are commonly used in the majority of PDE systems. For instance, in one-dimensional PDE systems, the most prevalent spatial partial derivatives are $\{\partial_x \boldsymbol{s}, \partial_{xx}\boldsymbol{s}, \partial_{xxx}\boldsymbol{s}, \ldots\}$, and the highest order d of the partial derivatives is typically chosen to be a small finite value (we set it to 4 in our experiments in this paper). Therefore, for a PDE system with a spatial dimension of D, the number of possible spatial derivative terms satisfies the following recursive formula:

$$S_D(d) = d + 1, \quad D = 1$$
$$S_D(d) = \sum_{i=0}^{d} S_{D-1}(i), \quad D > 1$$

where $S_D(d)$ represents the total number of potential derivative terms in a PDE system with a spatial dimension of D and a maximum derivative order of d.

It should be noted that these spatial partial derivatives are inherent to almost all PDE systems and can be estimated directly from observational data. Therefore, we do not introduce any additional specific prior knowledge.

C.2. Experimental results for DR and KS systems

First, we consider the DR system with control variable u. The mathematical formulation can be found in Table 2 of the main text. Here, we consider $u(x,t) = \sin(2\pi x/L_{DR}) + u_2(t)$, and $u_2(t)$ is a sampling function from GRF. After training using the FNODEs method, the experimental results are shown in Fig. S1. It is evident that our method exhibits high training efficiency and satisfactory predictive accuracy.

Similarly, we conducted experiments in the KS system using the function form described in Table 2 of the main text. Here, we consider $u(x,t) = \sin(2\pi x/L_{KS}) + u_3(t)$, and $u_3(t)$ is a sampling function from GRF. After training, the experimental results are shown in Figure S2. It is evident that our method also exhibits excellent predictive performance in the KS system.

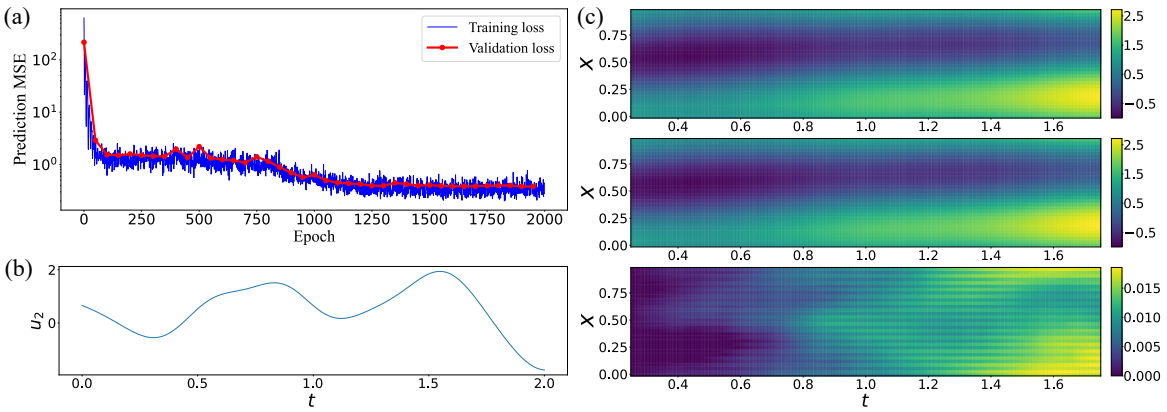

Figure S1. The experimental results of the DR system using FNODEs Method. (a) The variation of the training and validation loss during the training process. (b) Plot of $u_2(t)$ from a test data. (c) These three subplots from top to bottom represent the ground truth, predicted results, and prediction errors of test data.

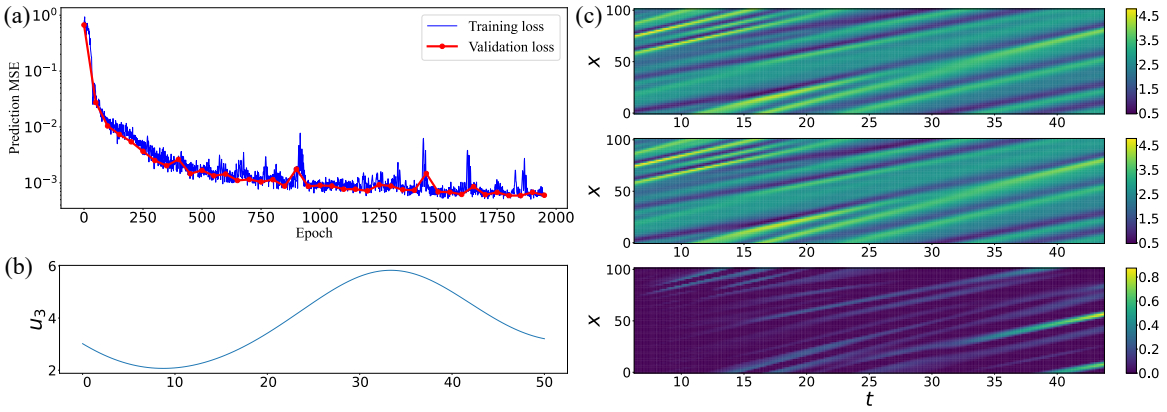

Figure S2. The experimental results of the KS system using FNODEs Method. (a) The variation of the training and validation loss during the training process. (b) Plot of $u_3(t)$ from a test data. (c) These three subplots from top to bottom represent the ground truth, predicted results, and prediction errors of test data.

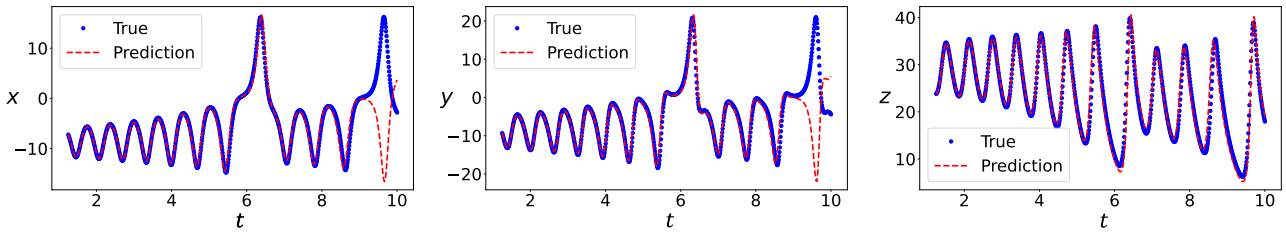

Figure S3. The prediction results of a test data in Lorenz system using FNODEs Method.

C.3. Experimental results on a chaotic system

Here, we consider a 3-D ODE system, known as the Lorenz63 system, which exhibits chaotic behavior. The system is described by the following equations:

$$\begin{aligned}\dot{x} &= 10(y - x),\\ \dot{y} &= \rho(t)x - y - xz,\\ \dot{z} &= xy - 8/3z,\end{aligned}$$

where $\rho(t)$ is a time-varying system parameter sampled from a Gaussian Random Field (GRF). Following the methods described in the main text, the training was completed in just 64.78 seconds for a total of 20,000 epochs. The experimental results are presented in Fig. S3. It is evident that our method demonstrates pretty good reconstruction performance on the chaotic Lorenz system.

D. Baseline methods

In this section, we provide a brief overview of the application of baseline methods in our experiments.

Firstly, in Section 4.2 of the main text, we utilize DeepONet as a baseline approach (Lu et al., 2019). This method comprises a branch net and a trunk net. The branch net receives m equidistantly sampled points from the parameter function $u(\boldsymbol{x}, t)$ as input and outputs a p-dimensional vector $\boldsymbol{b} = \{b_1, \cdots, b_p\}$. On the other hand, the trunk net takes the time t and space $\boldsymbol{x}$ as input and outputs a p-dimensional vector $\boldsymbol{t} = \{t_1, \cdots, t_p\}$. Then, the system state of any input variable can be predicted as follows:

$$\mathscr{G}(u)(\boldsymbol{x}, t) = \sum_{k=1}^{p} b_k(u) t_k(\boldsymbol{x}, t) + q,$$

where $q \in \mathbb{R}$ is a bias. After training, we can predict the state value $s(x, t)$ under any sampling function u.

Then, we consider the Fourier Neural Operator (FNO) (Li et al., 2020) method, which formulates a neural operator by directly parameterizing the integral kernel in Fourier space. In practice, we take the system state $s(\boldsymbol{x}, t)$ and the parameter function $u(\boldsymbol{x}, t)$ at time t as input and directly output the system state $s(\boldsymbol{x}, t + \delta_t)$ at time $t + \delta_t$. The FNO and DeepONet methods, as two neural operator approaches, effectively learn the prediction from the parameter function $u(t)$ to the system state. However, these methods lack interpretability and struggle to achieve better performance in extrapolation tasks beyond the distribution. Particularly, they exhibit poor performance in NS experiments where the initial values also vary.

Thirdly, we consider the PDE-NET (Long et al., 2018) method, which leverages finite differences to

approximate spatial derivative terms and uses a simple backward Euler for training and testing. In particular, for 2-d PDE systems, this method employs specific convolution kernels to compute spatial derivatives. In fact, this approach can be regarded as a numerical estimation of spatial partial derivatives and Euler iteration in the temporal direction. Although this is also an approach to estimating the gradient flow, in our experiments, the accuracy and speed of this method were inferior to the performance of the Fourier method.

Fourthly, we consider the NODEs for PDE systems. Indeed, due to the spatial derivatives of the PDE system, its dynamical behavior becomes highly intricate, posing numerous challenges in directly predicting the PDE system using NODEs. To execute the NODE method, we take the system state s and the parameter u as inputs to a neural network and obtain the temporal gradient of the system.

Fifthly, we contemplate utilizing the Physics-Informed Neural Networks (PINN) method (Raissi et al., 2019) for the experiment. Unlike other methods, this approach does not train the neural network based on the prediction error of observational data. The method operates on the premise of known dynamical equations and employs a neural network to directly fit the initial conditions, boundary conditions, and dynamical equations. However, this approach cannot directly model a family of dynamical systems. Therefore, for each test data, the neural network must be retrained for dynamics prediction.

Sixthly, we consider the DeepAR method (Flunkert et al., 2017), a forecasting method based on auto-regressive recurrent networks, which learns such a global model from historical data of all time series in the data set. However, our experimental results revealed that this method demonstrated poor performance in long-term prediction in differential equation systems, particularly in high-dimensional data within PDE systems. As a result, we only employ it as a baseline method for parametric ODE experiments.

Finally, we consider the message passing PDE solvers (MP-PDE) method (Brandstetter et al., 2022). This method models the grid as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $i \in V$, edges $ij \in \mathcal{E}$, and node features $\boldsymbol{f}_i \in \mathbb{R}^c$. Here, the feature $\boldsymbol{f}$ consists of the spatial derivative terms for finite difference method (FDM), finite volume method (FVM), and WENO schemes. Then, we can update the graph representation through message passing, which in turn enables us to predict the future state of the system.

In spite of the effectiveness of the aforementioned methods in addressing parametric PDE problems, there is still room for improvement in terms of learning accuracy and extrapolation capabilities. To address this, we introduce the FNODEs method, which demonstrates superior performance compared to baseline methods across various experimental scenarios.