Journal of Computational Physics 486 (2023) 112102

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Quadrature rule based discovery of dynamics by data-driven denoising ☆

Yiqi Gu [a], Michael K. Ng [b,*]

[a] School of Mathematical Sciences, University of Electronic Science and Technology of China, Sichuan 611731, China
[b] Department of Mathematics, The University of Hong Kong, Pokfulam, Hong Kong

ARTICLE INFO

Article history:
Received 14 September 2022
Received in revised form 21 March 2023
Accepted 26 March 2023
Available online 11 April 2023

Keywords:
Dynamical system
Deep learning
Quadrature rule
Denoising
Neural network

ABSTRACT

In this paper, we study the discovery of unknown dynamical systems with observed noisy data of the dynamics by neural networks. It is well-known that the performance of the neural network approach is degraded when observed data is noisy, even if the noise level is small. The main contribution of this paper is to propose a new network-based formulation for the dynamics discovery using numerical quadrature rules and to employ a self-supervision network to denoise observed data from the underlying dynamics. Our experimental results show that the performance of the proposed approach is better than that of existing dynamical discovery methods.

1. Introduction

Dynamical systems widely appear in physics, engineering, biology, etc. Discovering unknown dynamical systems using observed dynamics data is a significant and challenging problem. Early work of discovering dynamical systems [18,34] includes Gaussian processes [14,22,23,21], Symbolic regression [2,27], sparse regression [3,26,32,33] and statistical learning [18,34]. In recent years, neural network-based methods [25,17,16,31,9,20] are widely studied and particularly effective for high-dimensional systems. In particular, one uses a neural network to approximate the system's governing function and then discretize the dynamical system via special numerical methods. The approximate network is finally trained by minimizing a loss function derived from the discretized system. Such approaches trace back to the pioneering work [29], in which Runge-Kutta schemes were adopted for discretization. While some recent work focuses on the linear multistep methods (LMMs) [24,28,30]. In addition, using special physically relevant networks for approximation makes the discovered system more stable, and physically interpretable [31].

It is verified that neural network-based LMMs can identify complicated high-dimensional systems and have high approximation error orders for the governing function [7]. However, LMMs will lose their accuracy if the observed data is measured with noise. One typical example is the one-step Adams-Bashforth scheme, which computes the numerical derivative of the given state data. Even though the data is slightly perturbed by noise, its numerical derivative will be corrupted greatly, producing an intolerable error. Numerical experiments in [24] demonstrate that for small time step sizes, the accuracy of LMMs will be devastated by even 0.02% noise level with respect to the underlying dynamics.

☆ This work is supported by Hong Kong Research Grant Council GRF 12300218, 12300519, 17201020, 17300021, C1013-21GF, C7004-21GF and Joint NSFC-RGC N-HKU76921.

* Corresponding author.
E-mail addresses: yiqigu@uestc.edu.cn (Y. Gu), mng@maths.hku.hk (M.K. Ng).

https://doi.org/10.1016/j.jcp.2023.112102

In recent decades, the study of denoising has existed in many applications, especially in image denoising. A thorough review of image denoising techniques can be found in [12,6,19]. With the recent advances and applications of deep learning, neural network-based denoising methods have been rapidly developed. Fundamentally, the denoising process is characterized by a neural network (denoising network), which receives a corrupted image as an input and outputs a denoised clean image. In [15], a "Noise2Noise" self-supervision model is proposed to train the denoising network, in which one only needs ample corrupted images as a training dataset and does not need any explicit statistical likelihood model of the corruption or an image prior. In other words, the neural network denoising behaves as a "black box" and is totally driven by data. A recently enhanced "Neighbor2Neighbor" self-supervision model has been presented in [10], which only needs one corrupted copy for every image in the dataset; hence it is more applicable in practice.

In this paper, two contributions have been made to this work. First, we propose a new network-based formulation for the dynamics discovery relying on numerical quadrature rules. This quadrature rule-based formulation avoids computing the difference quotient of the data and hence is more robust than LMMs in noisy situations. Second, we develop a data-driven denoising technique for the dynamics data following the idea of the "Neighbor2Neighbor" self-supervision model. In the numerical experiments, several benchmark problems are solved. Compared with previous work [25,9,20] that solved problems with Gaussian noise up to 10% of the highest magnitude of the data, the proposed methods can easily handle 30% Gaussian noise. Also, we test temporally correlated noise and more complicated noise. The numerical results present the advantage of quadrature rule-based formulation over the existing LMMs and the superiority of the data-driven denoising.

This paper is organized as follows. In Section 2, we introduce the problem concerning the discovery of dynamics and discuss the details of the numerical quadrature approach. In Section 3, we review image denoising and propose the data-driven denoising technique for discovering dynamics. Numerical experiments are provided in Section 4 to validate the effectiveness of our methods. Finally, we conclude this paper in Section 5.

2. Discovery of dynamics

Let $d > 0$ be the dimension and $T > 0$ be the terminal time of the problem. We suppose that there exists a dynamical system with an initial condition, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{x}(t)), \quad 0 < t \le T, \tag{2.1}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_{\text{init}}, \tag{2.2}$$

where $\boldsymbol{x} : [0, T] \to \mathbb{R}^d$ is the state function; $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^d$ is the governing function; and $\boldsymbol{x}_{\text{init}} \in \mathbb{R}^d$ is some initial point from which the dynamics starts.

In many situations, neither $\boldsymbol{x}$ nor $\boldsymbol{f}$ is explicitly given. However, using technologies such as experimental observations and measurements, one can obtain a set of state data from the underlying dynamical system. Specifically, we let $N \in \mathbb{N}^+$ and $0 = t_0 < t_1 < \cdots < t_N = T$ be a sequence of time steps. Also, let $\tilde{\boldsymbol{x}}_n = \boldsymbol{x}(t_n) + \boldsymbol{\eta}_n$ be the observed state at time t_n, where $\boldsymbol{x}(t_n)$ is the ground truth and $\boldsymbol{\eta}_n$ is the noise term at time t_n. In this work, we assume $\boldsymbol{\eta}_0, \ldots, \boldsymbol{\eta}_N$ are i.i.d. random variables associated with some distribution $\mathcal{D}$. Then $\mathcal{X} := \{\tilde{\boldsymbol{x}}_n\}_{n=0}^N$ forms a dataset that is available to users. The objective of the dynamics discovery is identifying the governing function $\boldsymbol{f}$ provided that sufficiently many datasets are available.

2.1. Network-based linear multistep method

Recently, neural network-based approaches have been developed for the discovery of dynamics. The basic idea is using a neural network $\hat{\boldsymbol{f}} : \mathbb{R}^d \to \mathbb{R}^d$ to approximate $\boldsymbol{f}$ in the dynamical system (2.1)-(2.2). At the same time, the dynamical system is discretized by numerical methods, e.g., LMMs. For simplicity, in the following discussion, we assume that the time steps of the data are equidistant; namely, $t_n = nh$ for $n = 0, 1, \ldots, N$, where $h = T/N$. Then given a dataset $\mathcal{X} = \{\tilde{\boldsymbol{x}}_n\}_{n=0}^N$, (2.1) can be discretized by LMMs as follows:

$$\sum_{m=0}^{M} \alpha_m \tilde{\boldsymbol{x}}_{n-m} = h \sum_{m=0}^{M} \beta_m \hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{n-m}), \quad n = M, M+1, \ldots, N, \tag{2.3}$$

where $M \in \mathbb{N}^+$ is the number of LMM steps, $\alpha_m, \beta_m \in \mathbb{R}$ are specified coefficients and α_0 is always nonzero. Common types of LMMs include Adams-Bashforth (A-B) schemes, Adams-Moulton (A-M) schemes, and Backwards Differentiation Formula (BDF) schemes.

If some network $\hat{\boldsymbol{f}}$ satisfies the discretized system (2.3), it is an approximation of $\boldsymbol{f}$. Especially in the noise-free case, it holds that $\|\hat{\boldsymbol{f}} - \boldsymbol{f}\| \sim O(h^p)$, where p is the local truncation error order of the LMM scheme (see [7]). In practice, we can not find such a network $\hat{\boldsymbol{f}}$ that (2.3) is exactly satisfied, and hence one usually solve (2.3) by the following least squares model:

$$\min_{\hat{\boldsymbol{f}} \in \mathcal{A}} \frac{1}{N - M + 1} \sum_{n=M}^{N} \left| \sum_{m=0}^{M} \beta_m \hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{n-m}) - \sum_{m=0}^{M} h^{-1} \alpha_m \tilde{\boldsymbol{x}}_{n-m} \right|^2, \tag{2.4}$$

where the hypothesis space $\mathcal{A}$ is a class of candidate networks. The minimization (2.4) can be solved by standard optimizers (e.g., gradient descent method) under the deep learning framework.

2.2. Numerical quadrature

For a noisy-free situation, the effectiveness of the network-based LMM has been validated in previous literature [24,28,30]. However, the accuracy will deteriorate to a great extent, even with slightly noised data. For example, the A-B scheme with $M = 1$ (AB1) is exactly the forward Euler method, by which (2.3) reads

$$\frac{\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{x}}_{n-1}}{h} = \hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{n-1}), \quad n = 1, \dots, N, \tag{2.5}$$

where the left-hand side is the numerical differentiation of $\boldsymbol{x}$ at $t = t_n$. In this case, the noise of $\tilde{\boldsymbol{x}}_n$ and $\tilde{\boldsymbol{x}}_{n-1}$ will be amplified significantly when computing the difference quotient $(\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{x}}_{n-1})/h$ [4]. The same issue applies to other LMMs, such as the A-M scheme with $M = 1$ (AM1), which is given by

$$\frac{\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{x}}_{n-1}}{h} = \frac{1}{2}\left(\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{n-1}) + \hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_n)\right), \quad n = 1, \dots, N, \tag{2.6}$$

where the left hand side is the numerical differentiation of $\boldsymbol{x}$ at the half time step $t = t_{n-1/2}$.

A strategy is proposed to overcome the issue caused by noisy data. We integrate both sides of the dynamical system (2.1) from t_0 to t_n, which leads to

$$\boldsymbol{x}(t_n) - \boldsymbol{x}(t_0) = \int_{t_0}^{t_n} \boldsymbol{f}(\boldsymbol{x}(t))\mathrm{d}t. \tag{2.7}$$

Numerical quadrature rules can discretize the right-hand side of (2.7). For example, using the composite Trapezoidal rule, we obtain

$$\begin{aligned} \boldsymbol{x}(t_n) - \boldsymbol{x}(t_0) &= h\sum_{i=0}^{n-1}\left(\frac{1}{2}\boldsymbol{f}(\boldsymbol{x}_i) + \frac{1}{2}\boldsymbol{f}(\boldsymbol{x}_{i+1})\right) \\ &= h\left(\frac{1}{2}\boldsymbol{f}(\boldsymbol{x}_0) + \sum_{i=1}^{n-1}\boldsymbol{f}(\boldsymbol{x}_i) + \frac{1}{2}\boldsymbol{f}(\boldsymbol{x}_n)\right). \end{aligned} \tag{2.8}$$

Therefore, given a dataset $\mathcal{X} = \{\tilde{\boldsymbol{x}}_n\}_{n=0}^N$, the formulation to compute $\hat{\boldsymbol{f}}$ based on composite Trapezoidal rule directly follows (2.8) that

$$\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{x}}_0 = h\left(\frac{1}{2}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_0) + \sum_{i=1}^{n-1}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) + \frac{1}{2}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_n)\right), \quad n = 1, \dots, N. \tag{2.9}$$

It is clear that (2.9) is mathematically equivalent to the AM1 scheme (2.6). However, if n is moderately large, the numerical behaviors of (2.9) and (2.6) are pretty different. We do not need to compute any difference quotients in (2.9). The time step size h is integrated into the numerical quadrature rules on the right-hand side. So, on the one hand, the data noise on the left will not be amplified by being divided by h. On the other hand, the noise on the right will be averaged by multiplying by h. We should mention that this comparison is fair because both sides of (2.9) and (2.6) are bounded terms of $O(1)$ concerning N.

Similarly, using the composite Simpson's rule leads to the following formulation:

$$\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{x}}_0 = h\sum_{i=0}^{\lfloor n/2\rfloor - 1}\left(\frac{1}{3}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{2i}) + \frac{4}{3}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{2i+1}) + \frac{1}{3}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{2i+2})\right) + \delta_n \cdot h\left(\frac{1}{2}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{n-1}) + \frac{1}{2}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_n)\right), \quad n = 1, \dots, N, \tag{2.10}$$

where $\delta_n = 1$ if n is odd and $\delta_n = 0$ if n is even. The second term on the right-hand side of (2.10) means to use the Trapezoidal rule in the last subinterval.

In practice, we can seek $\hat{\boldsymbol{f}}$ by formulating the algebraic systems (2.9) and (2.10) into least squares optimization problems. Usually, one places the loss function by equating the coefficients of the squared terms to avoid any bias in training samples. By this principle, we propose the following least squares models.

- Trapezoidal rule

$$\min_{\hat{\boldsymbol{f}}\in\mathcal{A}} \frac{1}{N}\sum_{n=1}^{N}\left|\frac{1}{2n}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_0) + \frac{1}{n}\sum_{i=1}^{n-1}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) + \frac{1}{2n}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_n) - \frac{\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{x}}_0}{nh}\right|^2; \tag{2.11}$$

- Simpson's rule

$$\min_{\hat{\boldsymbol{f}}\in\mathcal{A}} \frac{1}{N}\sum_{n=1}^{N}\left|\frac{1}{\lfloor n/2\rfloor+\delta_n}\sum_{i=0}^{\lfloor n/2\rfloor-1}\left(\frac{1}{3}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{2i})+\frac{4}{3}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{2i+1})+\frac{1}{3}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{2i+2})\right)\right.$$
$$\left.+\frac{\delta_n}{\lfloor n/2\rfloor+\delta_n}\left(\frac{1}{2}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{n-1})+\frac{1}{2}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_n)\right)-\frac{\tilde{\boldsymbol{x}}_n-\tilde{\boldsymbol{x}}_0}{(\lfloor n/2\rfloor+\delta_n)h}\right|^2. \quad (2.12)$$

When n is moderately large, the quadrature-based formulation can avoid numerical differentiation of the data, which is very sensitive to noise. The numerical experiments in Section 4 also demonstrate the advantage of this method over the existing LMMs.

2.3. Further analysis and discussion

We give a conceptual analysis of the above methods. For simplicity, we assume that the hypothesis space of neural networks $\mathcal{A}$ is sufficiently broad such that zero training loss can be achieved. It is usually true if the candidate networks are over-parameterized and the free parameters are more than the squared terms. We do not consider the other case because it heavily relies on the approximation theory of neural networks, which is out of the scope of this paper.

Note that in supervised learning with noisy labels, if a solution achieves zero training loss, the error on the training set will be entirely governed by the noisiness of labels. Similarly, in this work, it suffices to study how the labels are corrupted by noise, which directly contributes to the training error of the solution. Therefore, we estimate the influence of the data noise $\{\boldsymbol{\eta}_n\}$ on the accuracy of LMMs and quadrature-based methods by investigating the extent to which $\{\boldsymbol{\eta}_n\}$ corrupts the true labels in the supervised learning model. We further assume that $\boldsymbol{\eta}_0,\dots,\boldsymbol{\eta}_N$ are bounded random variables that will not be very far away from the ground truth. Moreover, if the noise is unbounded but has no fat tails, the following result will still be valid with high probability.

For the LMMs, we study the AM1 scheme (2.6) as a typical example, whose least squares model is given by

$$\min_{\hat{\boldsymbol{f}}\in\mathcal{A}} \frac{1}{N}\sum_{n=1}^{N}\left|\frac{1}{2}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_{n-1})+\frac{1}{2}\hat{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_n)-\frac{\tilde{\boldsymbol{x}}_n-\tilde{\boldsymbol{x}}_{n-1}}{h}\right|^2. \quad (2.13)$$

Recall that $\tilde{\boldsymbol{x}}_n=\boldsymbol{x}(t_n)+\boldsymbol{\eta}_n$. If $\mathcal{A}$ is a subset of $[C^1(\mathbb{R}^d)]^d$ (it holds if the activation function of neural networks is C^1), then by Taylor's expansion, the above loss function can be written as

$$\frac{1}{N}\sum_{n=1}^{N}\left|\frac{1}{2}\hat{\boldsymbol{f}}(\boldsymbol{x}(t_{n-1}))+\frac{1}{2}\hat{\boldsymbol{f}}(\boldsymbol{x}(t_n))-\left(\frac{\boldsymbol{x}(t_n)-\boldsymbol{x}(t_{n-1})}{h}+I_n\right)\right|^2, \quad (2.14)$$

with

$$I_n:=-\frac{1}{2}\left(\nabla\hat{\boldsymbol{f}}_{n-1}\right)\boldsymbol{\eta}_{n-1}-\frac{1}{2}\left(\nabla\hat{\boldsymbol{f}}_n\right)\boldsymbol{\eta}_n+\frac{\boldsymbol{\eta}_n-\boldsymbol{\eta}_{n-1}}{h}, \quad (2.15)$$

$$\nabla\hat{\boldsymbol{f}}_n:=\left[\nabla\hat{f}_1(\boldsymbol{\xi}_{n,1})\ \nabla\hat{f}_2(\boldsymbol{\xi}_{n,2})\ \dots\ \nabla\hat{f}_d(\boldsymbol{\xi}_{n,d})\right]^\top, \quad (2.16)$$

where $\hat{f}_i$ is the i-the component of $\hat{\boldsymbol{f}}$, and $\boldsymbol{\xi}_{n,i}$ is some vector between $\boldsymbol{x}(t_n)$ and $\boldsymbol{x}(t_n)+\boldsymbol{\eta}_n$ entrywise, for any $1\le i\le d$. On the other hand, in the noise-free situation, the least squares loss function of the AM1 scheme is exactly (2.14) with I_n vanishing. Hence by adding the noise, the training label changes from $(\boldsymbol{x}(t_n)-\boldsymbol{x}(t_{n-1}))/h$ to $(\boldsymbol{x}(t_n)-\boldsymbol{x}(t_{n-1}))/h+I_n$.

Since the dataset $\{\tilde{\boldsymbol{x}}_n\}$ is distributed in a bounded domain, the C^1 smoothness of $\hat{\boldsymbol{f}}$ implies that $\|\nabla\hat{\boldsymbol{f}}_n\|_\infty$ is finite and uniformly bounded for all n. Also note that $h=T/N$, so the variation of the label is estimated as

$$I_n\sim O(N\boldsymbol{\eta}_{n-1}+N\boldsymbol{\eta}_n). \quad (2.17)$$

Next, we study the Trapezoidal rule (2.11) as a typical example of the quadrature-based methods. By a similar argument, we can write the loss function as

$$\frac{1}{N}\sum_{n=1}^{N}\left|\frac{1}{2n}\hat{\boldsymbol{f}}(\boldsymbol{x}(t_0))+\frac{1}{n}\sum_{i=1}^{n-1}\hat{\boldsymbol{f}}(\boldsymbol{x}(t_i))+\frac{1}{2n}\hat{\boldsymbol{f}}(\boldsymbol{x}(t_n))-\left(\frac{\boldsymbol{x}(t_n)-\boldsymbol{x}(t_0)}{nh}+J_n\right)\right|^2, \quad (2.18)$$

where the variation of the label

$$J_n=-\frac{1}{2n}\left(\nabla\hat{\boldsymbol{f}}_0\right)\boldsymbol{\eta}_0-\frac{1}{n}\sum_{i=1}^{n-1}\left(\nabla\hat{\boldsymbol{f}}_i\right)\boldsymbol{\eta}_i-\frac{1}{2n}\left(\nabla\hat{\boldsymbol{f}}_n\right)\boldsymbol{\eta}_n+\frac{\boldsymbol{\eta}_n-\boldsymbol{\eta}_0}{nh}. \quad (2.19)$$

We consider the case that n is moderately large such that $n = cN$ for some $0 \ll c \leq 1$. Since $\|\nabla \hat{\boldsymbol{f}}_i\|_\infty$ for every i is finite, by Lyapunov's central limit theorem, the distribution of $-\frac{1}{2n}\left(\nabla \hat{\boldsymbol{f}}_0\right)\boldsymbol{\eta}_0 - \frac{1}{n}\sum_{i=1}^{n-1}\left(\nabla \hat{\boldsymbol{f}}_i\right)\boldsymbol{\eta}_i - \frac{1}{2n}\left(\nabla \hat{\boldsymbol{f}}_n\right)\boldsymbol{\eta}_n$ approximates the normal distribution with mean 0 and variance σ^2/n if N is sufficiently large, where σ^2 is the variance of $\boldsymbol{\eta}_i$ for every i. Therefore this term is close to zero with high probability and hence dominated by $\frac{\boldsymbol{\eta}_n - \boldsymbol{\eta}_0}{nh}$ noting that $nh \sim O(1)$. Finally we have

$$J_n \sim O(\boldsymbol{\eta}_n + \boldsymbol{\eta}_0). \tag{2.20}$$

It is clear that J_n has a variance independent of N, but I_n has a variance quadratically amplified by N. This implies that the corruption of LMMs' labels will deteriorate quickly as N increases, but the corruption of quadrature-based methods' labels is always stable and will not be worse.

3. Network-based denoising

In the area of computer vision, self-supervised denoising techniques have been developed to handle noisy images [15,10]. Specifically, one introduces a neural network ϕ to serve as the denoising process. Let $\tilde{X}$ be a noisy observation of an image X, then $\phi(\tilde{X})$ is the denoised one. The denoising network ϕ is trained under a data-driven environment, where a dataset of ample observed images is provided. We adopt a similar idea in the discovery of dynamics, where we take the trajectories starting from a large number of initial points as the dataset.

3.1. Image denoising

We assume the observation noise is zero-mean. Suppose we are given a noisy observation $\tilde{X}$ of a target image X, then $\mathbb{E}(\tilde{X}) = X$. Without any extra information about X, the objective is to recover X as accurately as possible. Fortunately, a big dataset of noisy observations of other images with the same noise distribution is available. This recovery can be accomplished using the following network-based technique, which is proposed in [10]. Here, let us briefly describe the self-supervision learning technique. For simplicity, we assume all images have the same 2-D size $s_1 \times s_2$ with s_1, s_2 being even. We first introduce image pair sub-samplers (g_1, g_2) to generate noisy image pairs $(g_1(\tilde{X}), g_2(\tilde{X}))$ from a single noisy image $\tilde{X}$. We expect that the two sampled images $(g_1(\tilde{X}), g_2(\tilde{X}))$ closely resemble each other. We use $X_{i,j}$ to denote the (i, j)-th pixel of the image X. In [10], (g_1, g_2) is defined as follows:

$$g_1(\tilde{X})_{2j+1,2k+1} = g_1(\tilde{X})_{2j+1,2k+2} = g_1(\tilde{X})_{2j+2,2k+1} = g_1(\tilde{X})_{2j+2,2k+2} = Y_{j,k} \tag{3.1}$$

and

$$g_2(\tilde{X})_{2j+1,2k+1} = g_2(\tilde{X})_{2j+1,2k+2} = g_2(\tilde{X})_{2j+2,2k+1} = g_2(\tilde{X})_{2j+2,2k+2} = Z_{j,k} \tag{3.2}$$

for $j = 0, 1, \ldots, \frac{s_1}{2} - 1$ and $k = 0, 1, \ldots, \frac{s_2}{2} - 1$, where $(Y_{j,k}, Z_{j,k})$ is a randomly chosen pair of neighboring pixels in the 2×2 block $\{\tilde{X}_{2j+1,2k+1}, \tilde{X}_{2j+1,2k+2}, \tilde{X}_{2j+2,2k+1}, \tilde{X}_{2j+2,2k+2}\}$. Namely, g_1 takes one pixel, and g_2 takes another neighboring pixel to fill up all four pixels. In other words, g_1 and g_2 form a pair of neighborhood sub-samplers. For piecewise smooth images such as real-world pictures, it is clear that $g_1(\tilde{X}) \approx g_2(\tilde{X})$. Now the denoising network ϕ can be determined by the regularized optimization

$$\min_{\phi \in \mathcal{A}} \mathbb{E}_{X,\tilde{X}} \|\phi(g_1(\tilde{X})) - g_2(\tilde{X})\|_2^2 + \gamma \mathbb{E}_{X,\tilde{X}} \|\phi(g_1(\tilde{X})) - g_2(\tilde{X}) - g_1(\phi(\tilde{X})) + g_2(\phi(\tilde{X}))\|_2^2, \tag{3.3}$$

where $\gamma > 0$ is a regularization parameter. The first reconstruction term is used to force neural network ϕ to match $g_1(\tilde{X})$ and $g_2(\tilde{X})$, and the second regularization term is employed to match the gap between the ground truths of $g_1(\tilde{X})$ and $g_2(\tilde{X})$. In image denoising, one usually employs convolutional neural networks as the denoising network. The detailed explanation can be found in [10].

3.2. Denoising trajectories of the dynamics

Based on the denoising technique for images, we develop a special approach to denoise the dataset $\mathcal{X} = \{\tilde{\boldsymbol{x}}_n\}_{n=0}^N$ of a trajectory in the discovery of dynamics. For simplicity, we denote $\mathcal{X}^j = \{\tilde{x}_n^j\}_{n=0}^N$, where $\tilde{x}_n^j$ is the j-th component of $\tilde{\boldsymbol{x}}_n$, and discuss the denoising process for any one component of the state vectors. Without loss of generality, we assume N is odd. Then we propose a pair of neighbor sub-samplers (p_1, p_2) mapping a $(N+1)$-sequence to two $(N+1)$-sequences defined as

$$p_1(\mathcal{X}^j)_{2k} = \tilde{x}_{2k}^j, \quad p_1(\mathcal{X}^j)_{2k+1} = \begin{cases} \dfrac{\tilde{x}_{2k}^j + \tilde{x}_{2k+2}^j}{2}, & k < (N-1)/2 \\ \dfrac{-\tilde{x}_{2k-2}^j + 3\tilde{x}_{2k}^j}{2}, & k = (N-1)/2 \end{cases} \tag{3.4}$$

and

$$p_2(\mathcal{X}^j)_{2k+1} = \tilde{x}^j_{2k+1}, \quad p_2(\mathcal{X}^j)_{2k} = \begin{cases} \dfrac{3\tilde{x}^j_{2k+1} - \tilde{x}^j_{2k+3}}{2}, & k = 0 \\ \dfrac{\tilde{x}^j_{2k-1} + \tilde{x}^j_{2k+1}}{2}, & k > 0 \end{cases} \tag{3.5}$$

for $k = 0, \ldots, (N-1)/2$. Here p_1 preserves the even elements but redefines the odd elements by linear interpolation through the two neighboring elements (extrapolation for the endpoint). And p_2 operates in a converse way. Note that this manner is slightly different from (3.1)-(3.2), where unpreserved elements are forced to be the same as one neighboring element. We use linear interpolation for unpreserved elements because it makes $p_1(\mathcal{X}^j)$ closer to $\mathcal{X}^j$ so that less information will be lost.

Let ϕ_j be the denoising network for the j-th component, which is a mapping from $\mathbb{R}^{N+1}$ to $\mathbb{R}^{N+1}$. Then based on (3.3), we derive the following minimization:

$$\min_{\phi_j} \mathbb{E}_{x^j, \mathcal{X}^j} \|\phi_j(p_1(\mathcal{X}^j)) - p_2(\mathcal{X}^j)\|_2^2 + \gamma \mathbb{E}_{x^j, \mathcal{X}^j} \|\phi_j(p_1(\mathcal{X}^j)) - p_2(\mathcal{X}^j) - p_1(\phi_j(\mathcal{X}^j)) + p_2(\phi_j(\mathcal{X}^j))\|_2^2, \tag{3.6}$$

where the expectation is taken over the j-th component of the true trajectory x^j and the observed dataset $\mathcal{X}^j$.

Practically, a large dataset of observed trajectories is required to implement the preceding optimization. A straightforward way is choosing a large number of initial points and observing the trajectories that start from these points for $t \in (0, T]$. We use $\boldsymbol{x}(t; \boldsymbol{x}_{\text{init}})$ to denote the true trajectory starting from $\boldsymbol{x}_{\text{init}}$, whose j-th component is denoted as $x^j(t; \boldsymbol{x}_{\text{init}})$. Similarly, we use $\mathcal{X}(\boldsymbol{x}_{\text{init}})$ and $\mathcal{X}^j(\boldsymbol{x}_{\text{init}})$ to denote the sequences of the observed states and their j-th components, respectively. Suppose we have a dataset $\{\mathcal{X}(\boldsymbol{z}_m)\}_{m=1}^M$, then (3.6) can be implemented as

$$\begin{aligned} \min_{\phi_j \in \mathcal{A}} \frac{1}{M} \sum_{m=1}^{M} & \|\phi_j(p_1(\mathcal{X}^j(\boldsymbol{z}_m))) - p_2(\mathcal{X}^j(\boldsymbol{z}_m))\|_2^2 \\ & + \frac{\gamma}{M} \sum_{m=1}^{M} \|\phi_j(p_1(\mathcal{X}^j(\boldsymbol{z}_m))) - p_2(\mathcal{X}^j(\boldsymbol{z}_m)) - p_1(\phi_j(\mathcal{X}^j(\boldsymbol{z}_m))) + p_2(\phi_j(\mathcal{X}^j(\boldsymbol{z}_m)))\|_2^2, \end{aligned} \tag{3.7}$$

for all j. After training ϕ_j, we can regard $\phi_j(\mathcal{X}^j)$ as the denoised data of $\mathcal{X}^j$.

To denoise the entire dataset $\mathcal{X}$, one needs d independent neural networks ϕ_j, $j = 1, \ldots, d$, to deal with the d components of the data. We mention that every denoising network is only determined by the type of noise and independent of the data. Namely, the denoising network trained with one class of datasets could generalize for other datasets of the same type of noise.

We want to mention further that the denoising network operates as a "black box", which receives an original noisy dataset as its input and produces the corresponding denoised dataset as its output. Although we can compute the parameters (weights and biases, see Section 4.1) of the denoising network, it is hard to relate them with any physical meanings. However, the mechanism of denoising can be inferred partially from the learning dynamics of neural networks. In many situations, the network parameters are initialized as $O(1/\sqrt{W})$ with the same distribution (e.g., He initialization [8] and Lecun initialization [13]). On the other hand, in the neural tangent kernel regime [11,1], if the neural network is very wide, the variation of the weights/biases under gradient descent will be extremely small. Consequently, the network will nearly preserve a balanced distribution for all parameters during the training. In particular, the entries of the weight matrices share almost the same distribution, and they play the role of averaging out the input noise as the matrices are multiplied by the input vector. In this sense, the number of time steps (i.e., the length of the data sequence) should be sufficiently large so that the law of large numbers can eliminate the data noise.

As discussed in Section 2.3, the quadrature methods alleviate the noisiness by averaging out the i.i.d. noise on the data points. However, such averaging strategy is less effective if the noise is more complicated or not i.i.d. Fortunately, the denoising network is much more capable and can simulate many types of denoising. For example, the temporally correlated noise can be significantly reduced by the denoising networks (see Section 4.4.2). Therefore, we can use denoising networks to preprocess the datasets, removing any complicated components of the noise, and then adopt the quadrature methods to discover the unknown dynamics, in which the noise/error left by the denoising networks is refined by the averaging.

We also note that in the quadrature methods, the approximate network $\hat{\boldsymbol{f}}$ receives the denoised data as the input and produces the estimated dynamics as the output. So it is possible to combine the architectures of $\hat{\boldsymbol{f}}$ and the denoising networks $\{\phi_j\}$ as a composite neural network, which carries out both the tasks of denoising and discovery. Accordingly, one can develop an integrated loss function containing the ingredients of both denoising and discovery. Such an approach is likely more efficient than accomplishing the two tasks one by one separately, and is worth studying in the future.

4. Numerical examples

In this section, we implement the numerical quadrature technique for the discovery of dynamics proposed in Section 2.2. For comparison tests, the existing LMMs are also implemented. On the other hand, we implement the data-driven denoising technique proposed in Section 3.2 to deal with the datasets, compared with the case where no denoising is used.

4.1. Architecture of neural networks

In the numerical experiments, we employ the fully connected neural network (FNN) for the implementation. Specifically, given $L \in \mathbb{N}^+$ and $W \in \mathbb{N}^+$, which are the depth and width of the neural network, we define the nonlinear function $\boldsymbol{h}_\ell$ for $\ell = 1, \ldots, L-1$ as follows:

$$\boldsymbol{h}_\ell(\boldsymbol{z}_\ell) = \sigma\left(\boldsymbol{W}_\ell \boldsymbol{z}_\ell + \boldsymbol{b}_\ell\right), \tag{4.1}$$

where d is the dimension of the input and output; $\boldsymbol{W}_\ell \in \mathbb{R}^{W\times W}$, $\boldsymbol{b}_\ell \in \mathbb{R}^W$ for $\ell = 2, \ldots, L-1$ and $\boldsymbol{W}_1 \in \mathbb{R}^{W\times d}$, $\boldsymbol{b}_1 \in \mathbb{R}^d$; $\sigma(z)$ is the activation function, which is applied entry-wise to a vector. Common activation functions include rectified linear unit (ReLU) $\max\{0, z\}$ and the sigmoidal function $(1+e^{-z})^{-1}$. Then an FNN $\psi : \mathbb{R}^d \to \mathbb{R}^d$ is formulated as the composition of $\boldsymbol{h}_\ell$, $\ell = 1, \ldots, L-1$, namely,

$$\psi(\boldsymbol{z}) = \boldsymbol{A}^\top \boldsymbol{h}_{L-1} \circ \boldsymbol{h}_{L-2} \circ \cdots \circ \boldsymbol{h}_1(\boldsymbol{z}) \quad \forall \boldsymbol{z} \in \mathbb{R}^d, \tag{4.2}$$

where $\boldsymbol{A} \in \mathbb{R}^{d\times W}$. Note that for fixed d, L, W, σ, the FNN ψ is exactly determined by its parameters $\{\boldsymbol{A}, \boldsymbol{W}_\ell, \boldsymbol{b}_\ell : 1 \le \ell \le L-1\}$, which are trainable variables in deep learning. In the following experiments, both the approximate network $\hat{\boldsymbol{f}}$ and the denoising network ϕ are taken as the FNN architecture with ReLU activation.

4.2. Overall setting

The overall setting of the implementation is summarized as follows.

- *Environment.* The experiments are conducted in the Python environment. PyTorch library with CUDA toolkit is utilized for neural network implementation and GPU-based parallel computing.
- *Optimizers and hyper-parameters.* The network-involved optimization (2.4), (2.11), (2.12) and (3.7) are solved by the standard gradient descent method. We use 10^4 iterations for all examples. For the denoising optimization (3.7), we set the learning rate of the optimizer to decay from 10^{-4} to 10^{-5}; for the discovery of dynamics (2.4), (2.11) and (2.12), we set the learning rate to decay from 10^{-2} to 10^{-5}.
- *Network setting.* The denoising network is set as a ReLU FNN. We implement the denoising technique with various depths L and widths M to investigate their effects. Moreover, the approximate network $\hat{\boldsymbol{f}}$ is set as a ReLU FNN with fixed size $L = 5$ and $W = 640$, the optimal size among usual choices (Section 6.1.1 in [7]). Network parameters are initialized as

$$\boldsymbol{A}, \boldsymbol{W}_\ell, \boldsymbol{b}_\ell \sim U(-1/\sqrt{W}, 1/\sqrt{W}), \quad l = 1, \ldots, L-1. \tag{4.3}$$

- *Data generation.* In the example of the 2-D model problem, the state data is generated directly from the explicit expression of $\boldsymbol{x}$. In the other examples, no expression of $\boldsymbol{x}$ is available. Hence we generate the state data by solving the dynamical system using the subroutine `ode45` in Matlab with tiny tolerances (`RelTol`$= 10^{-13}$, `AbsTol`$= 10^{-13}$).
- *Testing set and error evaluation.* Given a noisy dataset $\mathcal{X} = \{\tilde{\boldsymbol{x}}_n\}_{n=0}^N$, we compute the following relative ℓ^2 error:

$$\text{data error} = \left(\frac{\sum_{n=0}^{N} \left\| \tilde{\boldsymbol{x}}_n - \boldsymbol{x}_n \right\|_2^2}{\sum_{n=0}^{N} \|\boldsymbol{x}_n\|_2^2} \right)^{\frac{1}{2}}, \tag{4.4}$$

which characterizes the noisiness of the original dataset. Similarly, we compute the error of the denoised dataset as

$$\text{denoising error} = \left(\frac{\sum_{n=0}^{N} \left\| \boldsymbol{\phi}(\mathcal{X})_n - \boldsymbol{x}_n \right\|_2^2}{\sum_{n=0}^{N} \|\boldsymbol{x}_n\|_2^2} \right)^{\frac{1}{2}}, \tag{4.5}$$

Table 4.1
Denoising errors for various noise types, depth L and width W. (OE: The original noise error of the dataset).

	$\sigma = 0.1$ (OE = 0.143)			$\sigma = 0.2$ (OE = 0.286)			$\sigma = 0.3$ (OE = 0.428)		
L	$W = 50$	$W = 100$	$W = 200$	$W = 50$	$W = 100$	$W = 200$	$W = 50$	$W = 100$	$W = 200$
2	1.520e-02	1.922e-02	1.929e-02	1.216e-02	1.811e-02	2.009e-02	**7.418e-03**	1.612e-02	1.868e-02
3	1.163e-02	1.030e-02	1.284e-02	1.510e-02	1.270e-02	1.713e-02	1.853e-02	1.242e-02	1.864e-02
4	8.633e-03	6.910e-03	8.718e-03	1.355e-02	1.151e-02	1.194e-02	1.912e-02	1.411e-02	1.460e-02
5	**4.251e-03**	6.352e-03	6.224e-03	**6.357e-03**	9.611e-03	1.035e-02	1.171e-02	1.331e-02	1.337e-02

(a) Gaussian noise

	$\sigma = 0.1$ (OE = 0.185)			$\sigma = 0.2$ (OE = 0.369)			$\sigma = 0.3$ (OE = 0.554)		
L	$W = 50$	$W = 100$	$W = 200$	$W = 50$	$W = 100$	$W = 200$	$W = 50$	$W = 100$	$W = 200$
2	1.802e-02	1.913e-02	2.198e-02	2.132e-02	2.045e-02	2.616e-02	2.624e-02	2.086e-02	2.870e-02
3	1.504e-02	1.110e-02	1.371e-02	2.049e-02	1.631e-02	1.857e-02	2.248e-02	1.989e-02	2.337e-02
4	8.812e-03	7.798e-03	1.080e-02	1.426e-02	1.274e-02	1.773e-02	1.782e-02	2.026e-02	1.984e-02
5	**5.793e-03**	8.218e-03	8.328e-03	**1.084e-02**	1.256e-02	1.735e-02	**1.718e-02**	2.020e-02	2.364e-02

(b) Gumbel noise

	$(\sigma, \beta) = (0.5, 0.1)$ (OE = 0.412)			$(\sigma, \beta) = (1.0, 0.2)$ (OE = 0.824)			$(\sigma, \beta) = (1.5, 0.3)$ (OE = 1.236)		
L	$W = 50$	$W = 100$	$W = 200$	$W = 50$	$W = 100$	$W = 200$	$W = 50$	$W = 100$	$W = 200$
2	1.584e-02	2.161e-02	1.964e-02	**1.235e-02**	2.628e-02	2.622e-02	1.428e-02	3.659e-02	3.684e-02
3	1.964e-02	1.853e-02	1.828e-02	3.146e-02	3.516e-02	3.087e-02	4.055e-02	4.937e-02	4.429e-02
4	2.319e-02	1.587e-02	1.820e-02	3.696e-02	2.792e-02	2.783e-02	5.404e-02	**3.455e-02**	3.687e-02
5	1.494e-02	**1.444e-02**	1.784e-02	3.204e-02	2.720e-02	3.505e-02	4.333e-02	3.696e-02	4.446e-02

(c) time-dependent noise

where $\boldsymbol{\phi}(\mathcal{X}) := \left[\phi_1(\mathcal{X}^1) \ \cdots \ \phi_d(\mathcal{X}^d)\right]^\top$. This denoising error characterizes the effectiveness of the denoising networks ϕ_j, $j = 1, \ldots, d$. Moreover, to estimate the performance of the discovery, we prescribe a set of N' time steps $\{t_n'\}_{n=1}^{N'} \subset [0, T]$ with uniform distribution as the testing set, and compute the following discrete relative ℓ^2 error:

$$\text{discovery error} = \left(\frac{\sum_{n=1}^{N'} \left\| \hat{\boldsymbol{f}}(\boldsymbol{x}(t_n')) - \boldsymbol{f}(\boldsymbol{x}(t_n')) \right\|_2^2}{\sum_{n=1}^{N'} \left\| \boldsymbol{f}(\boldsymbol{x}(t_n')) \right\|_2^2} \right)^{\frac{1}{2}}, \tag{4.6}$$

and where $\boldsymbol{x}(t_n')$ is computed either using the explicit expression or Matlab subroutine `ode45`. For all examples of discovery, we set N' as 10^4.

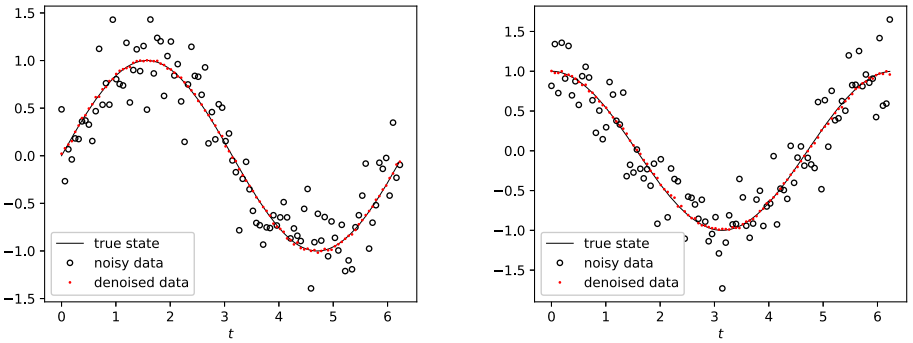
4.3. Denoising

We first investigate the performance of the network-based denoising technique proposed in Section 3.2 in dealing with datasets. Let us consider the 2-D model problem

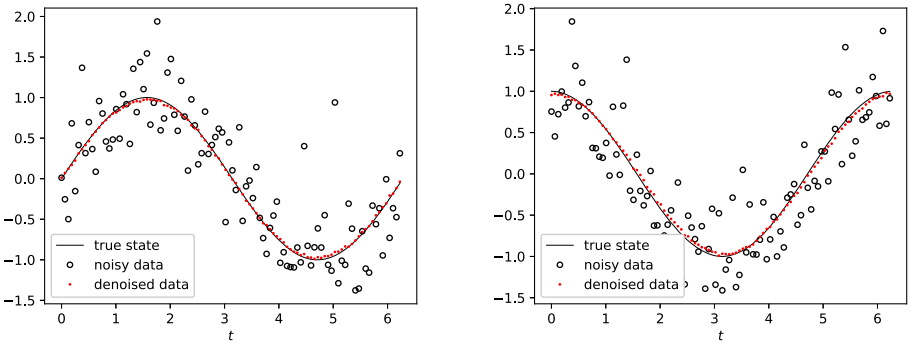
$$\begin{cases} \dfrac{\mathrm{d}x_1}{\mathrm{d}t} = x_2, \ \dfrac{\mathrm{d}x_2}{\mathrm{d}t} = -x_1, \quad t \in (0, 2\pi], \\ x_1(0) = \delta_1, \ x_2(0) = \delta_2, \end{cases} \tag{4.7}$$

with $\delta_1 = 0, \delta_2 = 1$. The true state function is $[x_1 \ x_2]^\top = [\sin(t) \ \cos(t)]^\top$. We generate the noisy dataset $\{\tilde{\boldsymbol{x}}_n\}_{n=0}^N$ with $N = 999$ from (4.7). The following three types of noise are tested in this example (η_n represents any component of the noise vector $\boldsymbol{\eta}_n$):

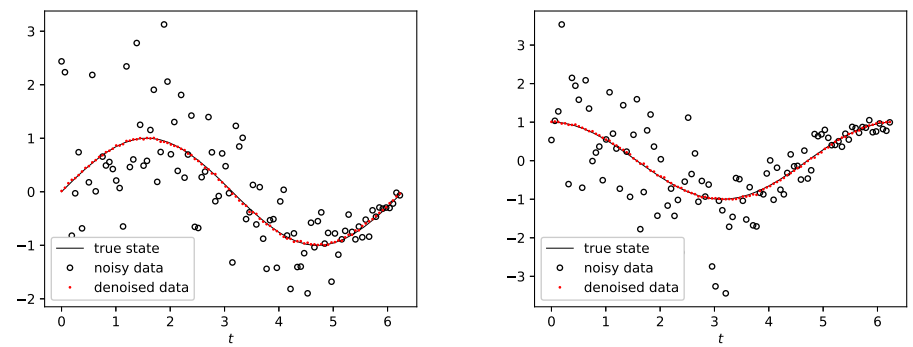
1. Gaussian noise: $\eta_n \sim \mathcal{N}(0, \sigma^2)$, where $\mathcal{N}(0, \sigma)$ is the normal distribution with mean zero and standard deviation σ. We set $\sigma = 0.1, 0.2, 0.3$.
2. Gumbel noise: $\eta_n \sim \mathcal{G}(\mu, \beta)$, where $\mathcal{G}(\mu, \beta)$ is the Gumbel distribution (the probability density function $p(x) = \beta^{-1} e^{-(x-\mu)/\beta} e^{-e^{-(x-\mu)/\beta}}$) with mean $\mu + \gamma\beta$ and standard deviation $\pi\beta/\sqrt{6}$. Here $\gamma = 0.57721\ldots$ is Euler's constant. We set $\beta = 0.1, 0.2, 0.3$ and $\mu = -\gamma\beta$, so the noise has zero means.



(a) Gaussian noise with $\sigma = 0.3$



(b) Gumbel noise with $\sigma = 0.3$



(c) time-dependent noise with $(\sigma, \beta) = (1.5, 0.3)$

Fig. 4.1. The true state function, original noisy data and denoised data of the model system (4.7). (Left: x_1; Right: x_2).

3. time-dependent noise: $\eta_n = (1 - t_n)\eta_n^{(1)} + t_n\eta_n^{(2)}$, where $\eta_n^{(1)} \sim \mathcal{N}(0, \sigma^2)$ and $\eta_n^{(2)} \sim \mathcal{G}(\mu, \beta)$. We set $(\sigma, \beta) = (0.5, 0.1), (1.0, 0.2), (1.5, 0.3)$ and $\mu = -\gamma\beta$. Note that the noise changes from large Gaussian noise to small Gumbel noise over time.

The noisy datasets and the true state functions are visualized in Fig. 4.1. Note that the Gaussian noise with $\sigma = 0.1/0.2/0.3$ is exactly 10%/20%/30% of the highest magnitude of the data. Comparatively, the previous work [25,9,20] solved similar problems with up to 10% Gaussian noise.

Next, we apply the denoising technique to deal with noisy data. Totally 10^3 initial points are uniformly generated in $[-1, 1] \times [-1, 1]$, whose trajectories are taken to form the training dataset of denoising. We test the effectiveness of various depth $L = 2, 3, 4, 5$ and width $W = 50, 100, 200$ of the denoising network. The denoising errors are shown in Table 4.1, and the denoised datasets are shown in Fig. 4.1. It is observed that the data errors are remarkably reduced after denoising

Table 4.2
Discovery errors using network-based LMMs (AB1,AB2,BDF1 and BDF2) and numerical quadrature (trapezoidal and Simpson's rules) of the model system (4.7).

	$\sigma = 0.1$		$\sigma = 0.2$		$\sigma = 0.3$	
Method	Noisy	Denoised	Noisy	Denoised	Noisy	Denoised
AB1	3.099e+00	9.996e-03	9.256e+00	2.316e-02	1.643e+01	1.629e-01
AB2	3.573e+00	1.137e-02	1.036e+01	5.389e-02	1.817e+01	4.715e-01
BDF1	3.746e+00	8.695e-03	9.501e+00	1.352e-02	1.859e+01	1.647e-01
BDF2	6.546e+00	1.081e-02	1.504e+01	3.284e-02	2.901e+01	4.358e-01
trapezoidal	3.175e-02	**6.610e-03**	5.065e-02	7.909e-03	7.997e-02	1.058e-02
Simpson	2.951e-02	6.653e-03	5.303e-02	**7.809e-03**	8.209e-02	**1.041e-02**

(a) Gaussian noise

	$\sigma = 0.1$		$\sigma = 0.2$		$\sigma = 0.3$	
Method	Noisy	Denoised	Noisy	Denoised	Noisy	Denoised
AB1	7.551e+00	2.427e-02	1.591e+01	1.329e-01	2.600e+01	2.007e+00
AB2	7.128e+00	2.981e-02	1.659e+01	4.042e-01	2.975e+01	1.585e+00
BDF1	5.094e+00	**1.088e-02**	1.383e+01	6.258e-02	2.483e+01	2.129e+00
BDF2	8.260e+00	1.519e-02	2.369e+01	4.672e-01	4.137e+01	3.550e+00
trapezoidal	4.618e-02	1.584e-02	1.098e-01	2.891e-02	1.794e-01	3.678e-02
Simpson	5.202e-02	1.582e-02	1.238e-01	**2.886e-02**	2.052e-01	**3.677e-02**

(b) Gumbel noise

	$(\sigma, \beta) = (0.5, 0.1)$		$(\sigma, \beta) = (1.0, 0.2)$		$(\sigma, \beta) = (1.5, 0.3)$	
Method	Noisy	Denoised	Noisy	Denoised	Noisy	Denoised
AB1	1.355e+01	2.568e-02	1.882e+01	1.929e+00	2.839e+01	2.273e+00
AB2	1.298e+01	5.439e-02	2.913e+01	1.412e+00	4.243e+01	1.577e+00
BDF1	1.977e+01	3.505e-02	5.175e+01	2.032e+00	6.479e+01	2.327e+00
BDF2	3.431e+01	8.580e-02	8.921e+01	2.943e+00	1.069e+02	3.501e+00
trapezoidal	1.380e-01	1.014e-02	2.446e-01	2.168e-02	3.939e-01	**3.057e-02**
Simpson	1.347e-01	**1.003e-02**	2.268e-01	**2.159e-02**	3.922e-01	3.084e-02

(c) time-dependent noise

from 0.1-1 to 0.001-0.01. We also find that $(L, W) = (5, 50)$ is the optimal choice for most cases, and wider networks with $W = 200$ do not perform better. Actually, as the width increases, the network training will be more difficult so that worse results are obtained.

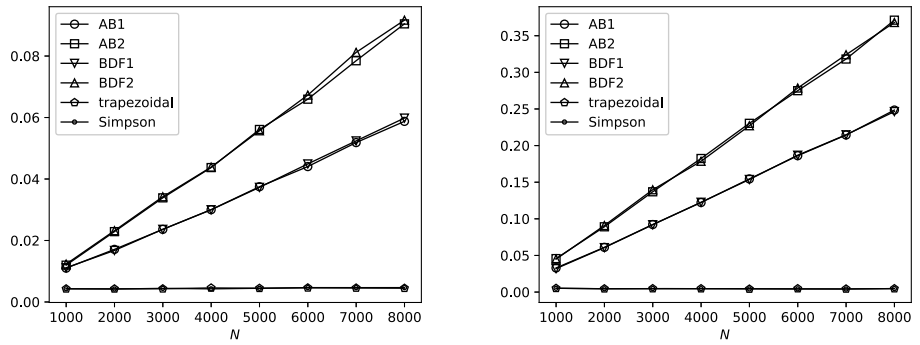
4.4. Discovery of dynamics

We next investigate the performance of the network-based numerical quadrature (trapezoidal rule (2.11) and Simpson's rule (2.12)) in the discovery of dynamics. We also implement the network-based LMMs (2.4) (A-B and BDF schemes of the first and second orders) for comparison. In these experiments, we use two types of datasets: the original noisy dataset and the denoised dataset. The denoised dataset is obtained by the denoising technique using denoising networks with $L = 5$ and $W = 50$.

4.4.1. Model problem

We continue to discover the model system (4.7) using the datasets generated and denoised from the preceding experiment in Section 4.3. For comparison, the discovery with the original datasets is also implemented. The discovery errors are reported in Table 4.2. It shows that (1) when the noise is relatively mild, for either LMMs or quadrature methods, the accuracy of the discovery with denoised datasets is remarkably better than that with original datasets; (2) if the noise is larger, LMMs collapse with noisy/denoised datasets, but quadrature methods still conserve $O(10^{-2})$ error levels with denoised datasets; (3) even with noisy datasets, quadrature methods can achieve errors from $O(10^{-2})$ to $O(10^{-1})$, but LMMs fail. This result implies that both the denoising technique and quadrature methods are highly effective for noisy problems.

Next, we test how the error changes for various N. As discussed in Section 2.3, the corruption caused by noise will be heavier as N increases for LMMs but not for quadrature methods. This fact could be reflected by their errors versus different N. In this test, we only use Gaussian noisy datasets with $\sigma = 0.005$ and 0.01 without denoising and take $N = 999, 1999, \ldots, 7999$. The discovery errors of LMMs and quadrature methods are plotted in Fig. 4.2. It is clear that the errors of LMM schemes are increasing with N, but those of quadrature methods keep almost unchanged. The numerical result is consistent with the theoretical discussion.



(a) $\sigma = 0.005$ (b) $\sigma = 0.01$

Fig. 4.2. Discovery errors for various N using network-based LMMs (AB1,AB2,BDF1 and BDF2) and numerical quadrature (trapezoidal and Simpson's rules) of the model system (4.7).

Table 4.3
Discovery errors using network-based LMMs (AB1,AB2,BDF1 and BDF2) and numerical quadrature (trapezoidal and Simpson's rules) of the Lorenz system (4.8).

	$\sigma = 0.3$		$\sigma = 0.4$		$\sigma = 0.5$	
Method	Noisy	Denoised	Noisy	Denoised	Noisy	Denoised
AB1	9.264e+01	6.293e-02	1.254e+02	1.655e-01	1.570e+02	1.462e-01
AB2	5.948e+01	5.259e-02	7.992e+01	1.046e-01	9.929e+01	1.068e-01
BDF1	9.231e+01	4.364e-02	1.254e+02	5.779e-02	1.572e+02	1.309e-01
BDF2	1.566e+02	7.297e-02	2.036e+02	2.130e-01	2.667e+02	2.493e-01
trapezoidal	2.987e+00	**3.111e-02**	2.174e+00	5.032e-02	2.118e+00	3.793e-02
Simpson	1.936e+00	3.498e-02	1.200e+00	**4.989e-02**	3.202e+00	**3.657e-02**

4.4.2. Lorenz system

We consider the following Lorenz system:

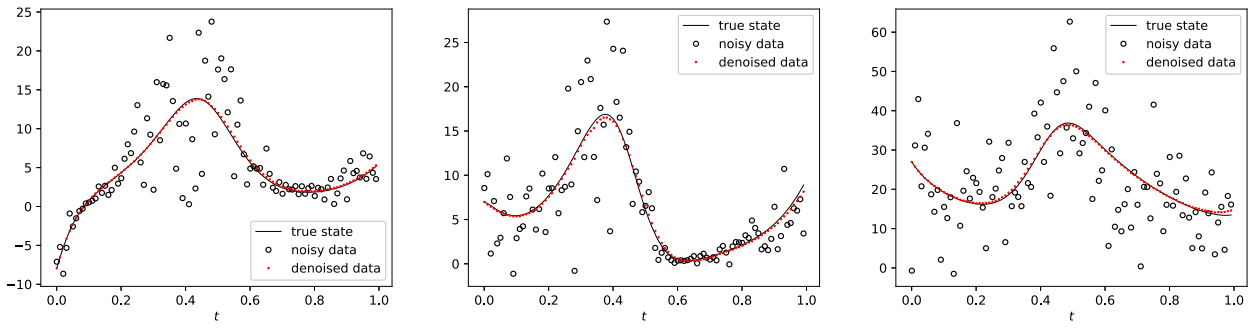
$$\begin{cases} \dfrac{dx_1}{dt} = 10(x_2 - x_1), \ \dfrac{dx_2}{dt} = x_1(28 - x_3) - x_2, \ \dfrac{dx_3}{dt} = x_1x_2 - 8x_3/3 \quad t \in (0, 1], \\ x_1(0) = -8, \ x_2(0) = 7, \ x_3(0) = 23, \end{cases} \tag{4.8}$$

which characterizes chaotic dynamics and has many applications, including weather forecasting. We set the number of time steps $N = 1999$. In this example, we use the following temporally correlated noise added to the ground truth $x(t_n)$ (x represents any component of $\boldsymbol{x}$):

$$\eta_n = x(t_n) \cdot \zeta_n \quad \text{with} \quad \zeta_n = \begin{cases} \xi + \frac{1}{2}\zeta_{n-1}, & n > 0, \\ \xi, & n = 0, \end{cases} \qquad \xi \in \mathcal{N}(0, \sigma^2); \tag{4.9}$$

namely, up to a multiplier $x(t_n)$, the noise at the current step is the sum of an independent Gaussian noise and half of the noise at the last step. We list the results of $\sigma = 0.3, 0.4, 0.5$ for a clear comparison.

We apply the denoising technique with initial points uniformly generated in $[-9, -7] \times [6, 8] \times [26, 28]$ forming the training set. The noisy/denoised data for $\sigma = 0.4$ are shown in Fig. 4.3. Next, the discovery error using LMMs and quadrature methods is reported in Table 4.3. It shows that for original noisy datasets, both quadrature methods and LMMs fail to work correctly. However, for denoised datasets, quadrature methods can perform effectively. It implies that the denoising network may remove complicated components of the noise that the quadrature methods can not handle. We plot the true governing function and discovered governing functions from noisy/denoised data in Fig. 4.4 to visualize the result. It can be observed that LMMs are numerically unstable, having significant erroneous spikes in some regions. But quadrature methods produce more stable results.



(a) x_1 (b) x_2 (c) x_3

Fig. 4.3. The noisy data and denoised data of the Lorenz system (4.8) (each data x has Gaussian noise with zero mean and standard deviation $0.3|x|$).

4.4.3. Glycolytic oscillator

Let us consider the 7-D model of oscillations in yeast glycolysis, which is given by

$$\begin{cases} \dot{S}_1 = J_0 - \dfrac{k_1 S_1 S_6}{1+(S_6/K_1)^q}, \\ \dot{S}_2 = 2\dfrac{k_1 S_1 S_6}{1+(S_6/K_1)^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5, \\ \dot{S}_3 = k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6), \\ \dot{S}_4 = k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7), \\ \dot{S}_5 = k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5, \\ \dot{S}_6 = -2\dfrac{k_1 S_1 S_6}{1+(S_6/K_1)^q} + 2k_3 S_3 (A - S_6) - k_5 S_6, \\ \dot{S}_7 = \psi\kappa (S_4 - S_7) - kS_7, \\ [S_1\ S_2\ S_3\ S_4\ S_5\ S_6\ S_7]^\top|_{t=0} = \boldsymbol{S}_0 := [1.125\ 0.95\ 0.075\ 0.16\ 0.265\ 0.7\ 0.092]^\top. \end{cases} \quad t \in [0,4]. \tag{4.10}$$

The model parameters are taken from Table 1 in [5].

The number of time steps is set as $N = 3999$. In this example, we artificially construct a polynomial-like type of noise that could imitate realistic noise. This choice is due to the good approximation properties of polynomials to common functions appearing in the real world. Specifically, the following compound noise is generated and added to the ground truth $x(t_n)$:

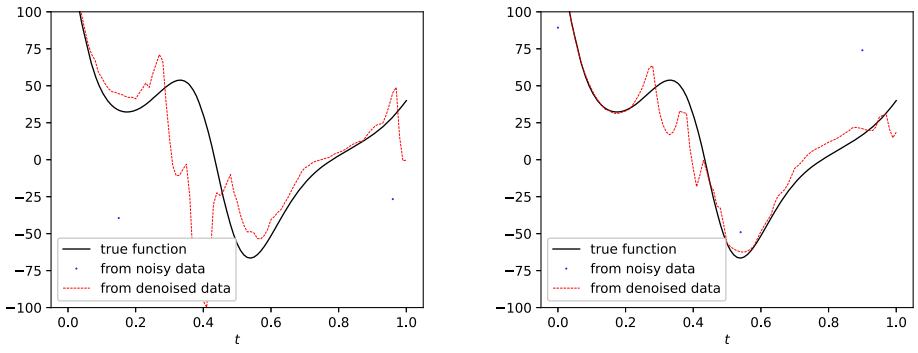
$$\eta_n = x(t_n) \cdot \sum_{k=0}^{K} \zeta_k \xi^k, \quad \xi \sim \mathcal{N}(0,\sigma), \quad \zeta_k \sim \mathcal{U}([-I, I]) \text{ for all } k, \tag{4.11}$$

where $\mathcal{U}([-I, I])$ is the uniform distribution in $[-I, I]$. The noise η in (4.11) is a polynomial of a Gaussian random variable with coefficients being uniform random variables. So when K is moderately large, the noise is quite complicated, and its probability density function is not straightforward to write down. In our experiment, we set $K = 3$ and various $(\sigma, I) = (0.05, 0.5), (0.1, 1.0), (0.15, 1.5)$. The trajectories from initial points uniformly generated in the box $\{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{S}_0\|_\infty \le 0.2\}$ are taken as the training set of the denoising. The noisy/denoised data are plotted in Fig. 4.5, where the compound noise is removed to a great extent by the denoising technique. Next, the true/discovered governing functions are shown in Fig. 4.6, and the discovery errors are reported in Table 4.4. The dynamics discovered by quadrature methods are observed to be much more accurate than those by LMMs, especially for the non-smooth spike parts of the governing function.

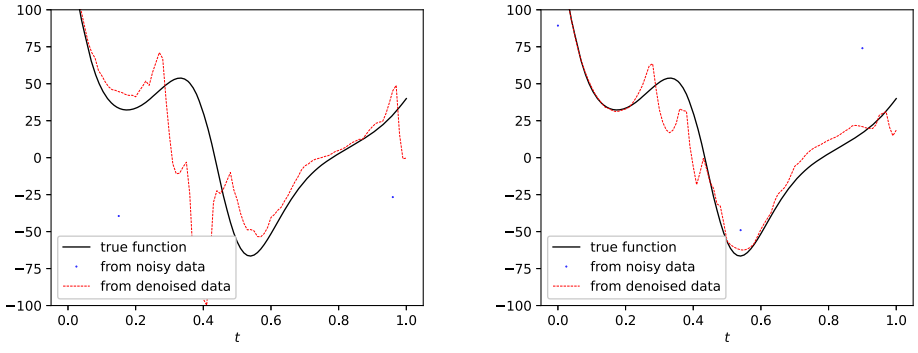
5. Conclusion

We develop two techniques to tackle the discovery of dynamical systems with noisy observed data. The first technique is the numerical quadrature-based formulation. A neural network is introduced to approximate the governing function and is trained using least squares optimization derived from quadrature rules. The second technique is the data-driven denoising method, in which another neural network is employed to simulate the denoising process that removes the noise of the data. Numerical experiments demonstrate that these techniques individually improve the accuracy compared with existing methods without denoising, and the best performance can be achieved by combining the two techniques.

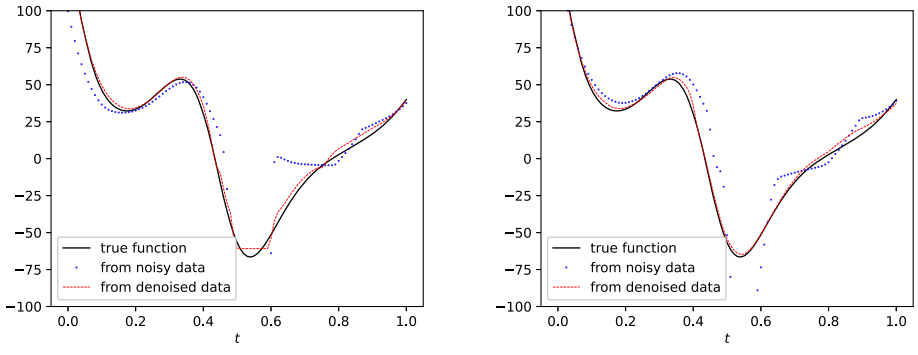
We mention that the proposed data-driven denoising method performs implicitly like a "black box" and can deal with data noise even if the noise is not analytic and implicit. Therefore, this method applies to practical problems in the real world. However, one limitation lies here every denoising network is only for one type of noise. This method will fail if the



(a) AB1 (b) AB2



(c) BDF1 (d) BDF2

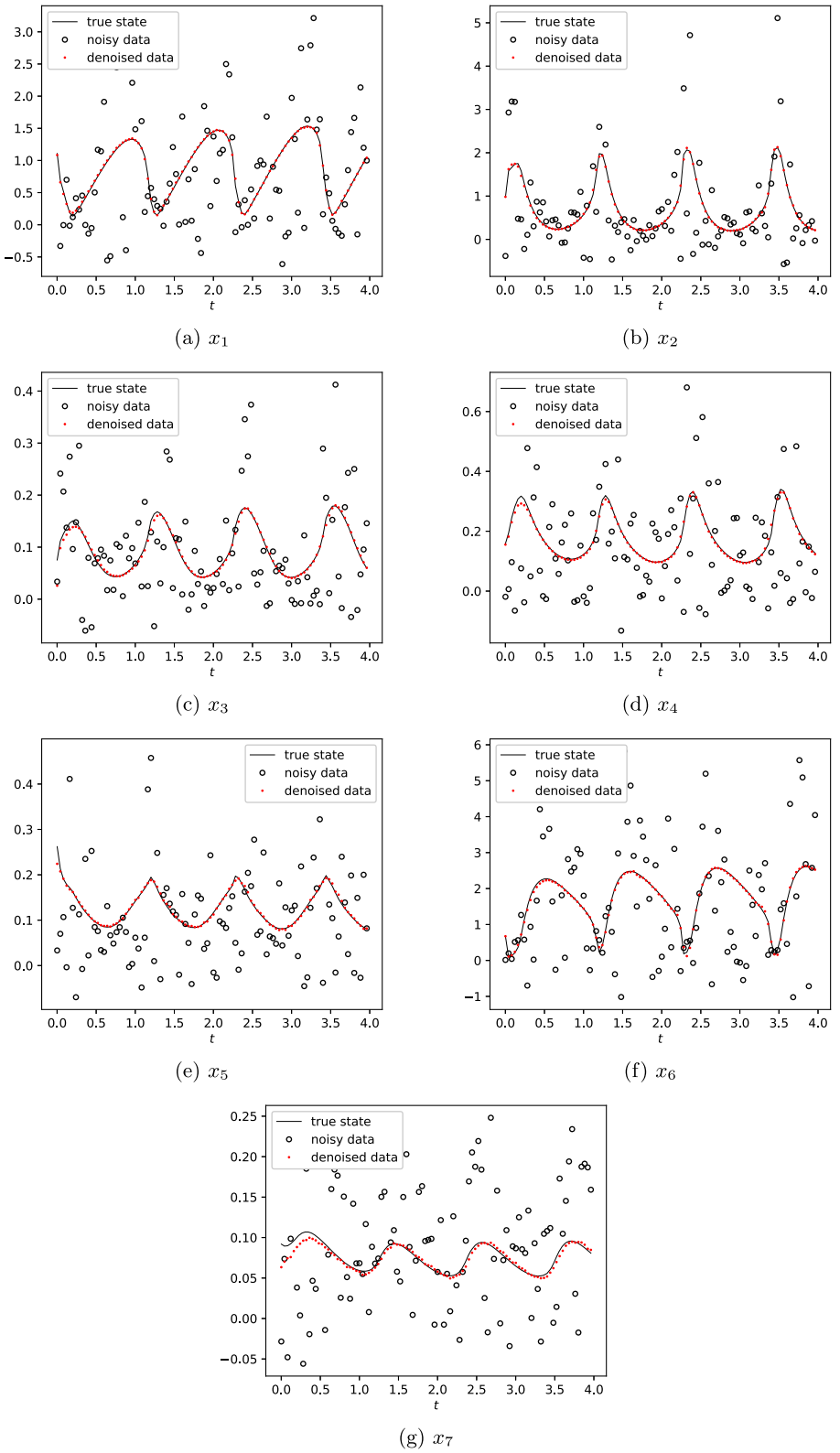


(e) trapezoidal (f) Simpson

Fig. 4.4. The true governing function and discovered governing function from noisy/denoised data of the Lorenz system (4.8).

Table 4.4
Discovery errors using network-based LMMs (AB1,AB2,BDF1 and BDF2) and numerical quadrature (trapezoidal and Simpson's rules) of the Glycolytic Oscillator (4.10).

	$(\sigma, I) = (0.05, 0.5)$		$(\sigma, I) = (0.1, 1.0)$		$(\sigma, I) = (0.15, 1.5)$	
Method	Noisy	Denoised	Noisy	Denoised	Noisy	Denoised
AB1	4.211e+02	3.740e+00	6.354e+02	1.638e+00	8.556e+02	1.139e+01
AB2	2.441e+02	2.390e+00	3.684e+02	1.067e+00	4.963e+02	6.242e+00
BDF1	4.199e+02	4.314e+00	6.336e+02	1.577e+00	8.534e+02	1.086e+01
BDF2	7.556e+02	7.277e+00	1.140e+03	2.847e+00	1.535e+03	2.097e+01
trapezoidal	1.052e+00	7.510e-02	2.015e+00	6.858e-02	3.243e+00	**1.117e-01**
Simpson	1.124e+00	**6.950e-02**	2.429e+00	**5.524e-02**	3.413e+00	1.253e-01



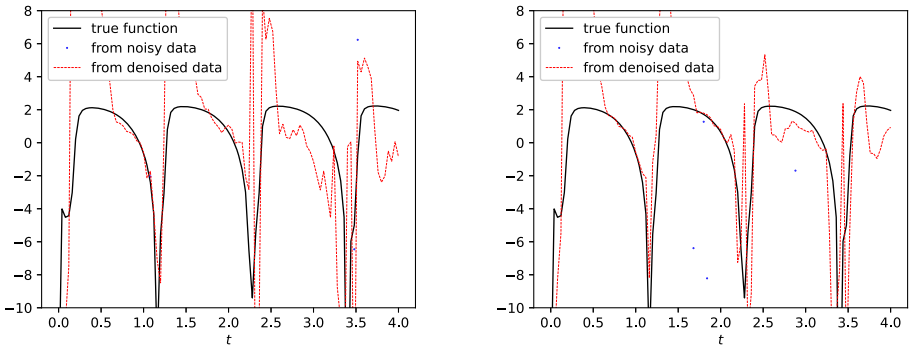
(a) x_1 (b) x_2

(c) x_3 (d) x_4

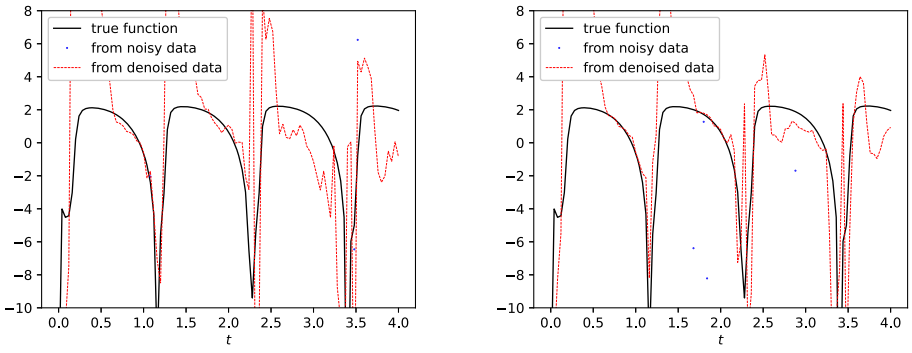
(e) x_5 (f) x_6

(g) x_7

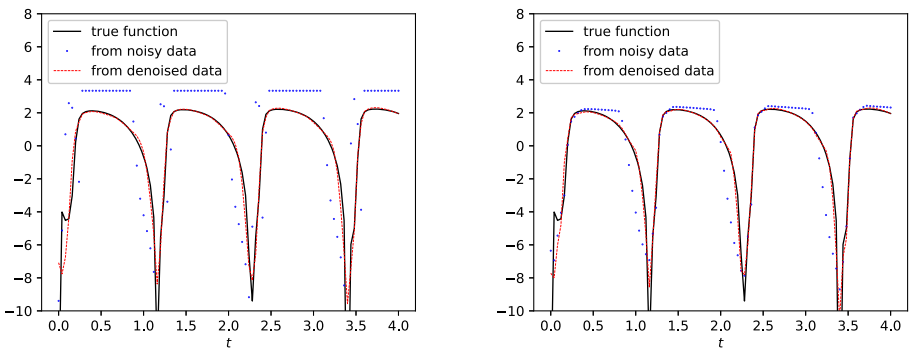
Fig. 4.5. The noisy data and denoised data of the Glycolytic Oscillator (4.10) (each data point x has compound noise (4.11) with $(\sigma, I) = (0.15, 1.5)$).



(a) AB1 (b) AB2



(c) BDF1 (d) BDF2



(e) trapezoidal (f) Simpson

Fig. 4.6. The true governing function and discovered governing function from noisy or denoised data of the Glycolytic Oscillator (4.10).

training dataset is noised from different sources. Unfortunately, a sufficiently large training set of the same noise type is sometimes unavailable. Also, since every denoising network has a fixed size, it only receives samples of the fixed dimension, which may cause inconvenience when noisy samples are of various sizes. Future work might improve the proposed method in overcoming these two limitations.

CRediT authorship contribution statement

Yiqi Gu: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Michael K. Ng:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] S. Arora, S.S. Du, W. Hu, Z. Li, On Exact Computation with an Infinitely Wide Neural Net, Advances in Neural Information Processing Systems, vol. 32, 2019.
[2] J. Bongard, H. Lipson, Automated reverse engineering of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 104 (2007) 9943–9948.
[3] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937.
[4] R. Chartrand, Numerical differentiation of noisy, nonsmooth data, Int. Sch. Res. Not. (2011).
[5] B.C. Daniels, I. Nemenman, Efficient inference of parsimonious phenomenological models of cellular dynamics using S-systems and alternating regression, PLoS ONE 10 (2015) e0119821.
[6] M. Diwakar, M. Kumar, A review on CT image noise and its denoising, Biomed. Signal Process. Control 42 (2018) 73–88.
[7] Q. Du, Y. Gu, H. Yang, C. Zhou, The discovery of dynamics via linear multistep methods and deep learning: error estimation, SIAM J. Numer. Anal. (2022).
[8] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on Imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015.
[9] P. Hu, W. Yang, Y. Zhu, L. Hong, Revealing hidden dynamics from time-series data by ODENet, J. Comput. Phys. 461 (2022) 111203.
[10] T. Huang, S. Li, X. Jia, H. Lu, J. Liu, Neighbor2Neighbor: self-supervised denoising from single noisy images, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
[11] A. Jacot, F. Gabriel, C. Hongler, Neural Tangent Kernel: Convergence and Generalization in Neural Networks, Advances in Neural Information Processing Systems, vol. 31, 2018.
[12] P. Jain, V. Tyagi, A survey of edge-preserving image denoising methods, Inf. Syst. Front. 18 (2016) 159–170.
[13] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-Normalizing Neural Networks, Advances in Neural Information Processing Systems, vol. 30, 2017.
[14] J. Kocijan, A. Girard, B. Banko, R. Murray-Smith, Dynamic systems identification with Gaussian processes, Math. Comput. Model. Dyn. Syst. 11 (2005) 411–424.
[15] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, T. Aila, Noise2Noise: learning image restoration without clean data, https://arxiv.org/abs/1803.04189, 2018.
[16] Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. 399 (2019) 108925.
[17] Z. Long, Y. Lu, X. Ma, B. Dong PDE-Net, Learning PDEs from data, in: International Conference on Machine Learning, 2018.
[18] F. Lu, M. Zhong, S. Tang, Nonparametric inference of interaction laws in systems of agents from trajectory data, Proc. Natl. Acad. Sci. 116 (29) (2019) 14424–14433.
[19] P. Milanfar, A tour of modern image filtering: new insights and methods, both practical and theoretical, IEEE Signal Process. Mag. 30 (2012) 106–128.
[20] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. 395 (2019) 620–635.
[21] M. Raissi, G.E. Karniadakis, Hidden physics models: machine learning of nonlinear partial differential equations, J. Comput. Phys. 357 (2018) 125–141.
[22] M. Raissi, P. Perdikaris, G.E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, J. Comput. Phys. 335 (2017) 736–746.
[23] M. Raissi, P. Perdikaris, G.E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, J. Comput. Phys. 348 (2017) 683–693.
[24] M. Raissi, P. Perdikaris, G.E. Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, https://arxiv.org/abs/1801.01236, 2018.
[25] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[26] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614.
[27] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (2009) 81–85.
[28] R. Tipireddy, P. Perdikaris, P. Stinis, A. Tartakovsky, A comparative study of physics-informed neural network models for learning unknown dynamics and constitutive relations, https://arxiv.org/abs/1904.04058, 2019.
[29] Y.-J. Wang, C.-T. Lin, Runge-Kutta neural network for identification of dynamical systems in high accuracy, IEEE Trans. Neural Netw. 9 (2) (1998) 294–307.
[30] X. Xie, G. Zhang, C.G. Webster, Non-intrusive inference reduced order model for fluids using deep multistep neural network, Mathematics 7 (8) (2019).
[31] H. Yu, X. Tian, W. E, Q. Li, OnsagerNet: learning stable and interpretable dynamics using a generalized Onsager principle, Phys. Rev. Fluids 6 (11) (2021) 114402.
[32] S. Zhang, G. Lin, Robust data-driven discovery of governing physical laws with error bars, Philos. Trans. R. Soc. A, Math. Phys. Eng. Sci. 474 (2217) (2018) 20180305.
[33] S. Zhang, G. Lin, SubTSBR to tackle high noise and outliers for data-driven discovery of differential equations, J. Comput. Phys. 428 (2021) 109962.
[34] M. Zhong, J. Miller, M. Maggioni, Data-driven discovery of emergent behaviors in collective dynamics, Phys. D, Nonlinear Phenom. 441 (2020) 132542.