Learning Nonlinear Dynamics from Experimental Data

Universal Differential Equations for Duffing-like System Identification in Presence of Noise

By XINYI WU

Supervised by DAVID BARTON

Department of Engineering Mathematics

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of MSC IN ENGINEERING MATHEMATICS in the Faculty of Engineering.

SEPTEMBER 2021

Abstract

As for a dynamical system of practical problems, there is generally abundant existed datasets obtained from the dynamical system. To predict its future performances and potential behaviours, the governing physical characteristics and fundamentally mathematical model is keen to be discovered for catching vital mechanical features and extrapolating accurate prediction. Based on mature ML techniques and advanced study on universal approximation for differential equations, it is available to reach the goal for directly studying a complicated system in the fields of nonlinear dynamics by experimental data without previously known knowledge. Meanwhile, besides guaranteeing sufficient accuracy, some complex properties of nonlinear dynamics system, such as chaotic and counterintuitive features, are allowed to be dealt with by numerical learnt simulation. As one of newly developing method in the fields of ML, universal differential equations model has given an excellent performance on numerical simulation based on differential models. In our work, we focus on utilize this technique to derive underlying scientific model from a specific dataset.

Dedication and acknowledgements

I would like to express my sincere appreciation to my supervisor Dr. David Barton for the supervision of the whole project and providing professional expertise. There is no doubt that I could not complete this project without his guidance. I would also like to thanks to Mr Sandor Beregi for his great help in academic discussions. It would be quite difficult to explore and do further research by myself on this topic with which I was not familiar. With their support, the summer project has been a memorable experience for me. Therefore, I want to express my deepest gratitude to them here.

Finally, to my family and friends, I would like to give my thanks for their emotional support. Due to the spread of COVID-19, the most of students, including me, had to stay at home and study online for the whole year, which was an unconventional learning experience, but made me feel puzzled and lonely sometimes. Hence, I want to show my great thanks to them here for who enlightened me during my Master's programme.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Postgraduate Programmes and that it has not been submitted for any other academic award.

Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such.

Any views expressed in the dissertation are those of the author.

SIGNED: Xinyi Wu ..... DATE: 15 September 2021 .....

TABLE OF CONTENTS

List of Tables

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

In this chapter, we will introduce the previous related work about our work and the research aim , which is divided into four parts. In Section 1.1, we briefly present historic research in the fields of nonlinear dynamics and neural networks which is the main academic background for our work. In Section 1.2, we will provide our problem statement and the corresponding research significance. In Section 1.3, we focus on describing the importance of our study beyond the academic context for some relevant stakeholders. In Section 1.4, the outline of this paper can be found.

1.1 Background

Research into dynamics has a long history. From the mid-17th century, the development of industry raised the demand of studying dynamical phenomena. Thus, academic research started to focus on mechanical and engineering systems. The general aim of modelling a dynamical system is to determine its potential mathematical structure and predict its future states, behaviours or possible evolution, such as propagation of solitary wave[19]. Typically, dynamical systems can be described by several differential equations over time[40], however, which are also usually complex, nonlinear and difficult to explore its functionally scientific model[59]. As research continued with more sophisticated systems, complicated mathematical and numerical techniques were required to structure nonlinear dynamics models and academically solve the corresponding dynamical problems in the forms of various differential equations[44]. Therefore, scholastic work with respect to numerical simulation and extrapolation on complex or even unknown nonlinear dynamics has created an explosion of interest[55].

In recent advances, based on abundant observed data collected from various sources, Machine Learning (ML) has been widely used in numerical experiments in order to learn expected

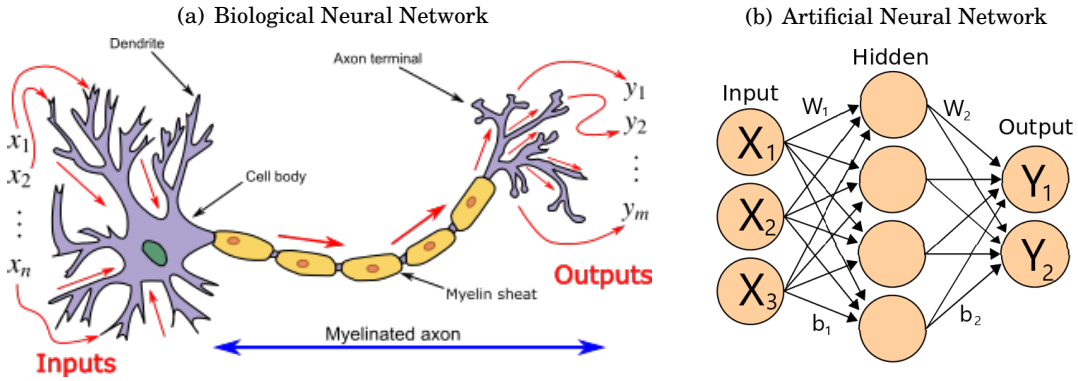


FIGURE 1.1. Introduction on Artificial neural network: (a)ANN is inspired by biological neural network in human brains[1]; (b)The simple explanation about how ANN works in computer through numerical algorithms.

engineering properties and make predictions for future tendency of the system[41]. Aiming to improve the efficiency and precision of identifying nonlinear dynamics, there have been lots of excellent methods for numerical analysis such as nonlinear auto-regressive models[8] and Gaussian Processes[56]. Artificial neural networks are one of the most popular approaches of ML in many application fields, which is a numerical technique structured by special network in computational systems, inspired by biological brains (see Figure 1)[9]. In the case of learning nonlinear dynamical models which have huge amounts of available data, artificial neural network is a more flexible and dependable tool[61], especially considering its efficiency and accuracy while contrasting with other kinds of numerical methods[63].

Even though modern ML techniques are likely to simulate some specific actual systems by learning given datasets, they require sufficient big data and computing time to promise the accuracy of expected numerical results and the convergence of experimental approximation. Meanwhile, they usually require careful selection from kinds of the network structures to achieve a good performance. Besides, the mathematical structure of nonlinear dynamics in the physical world is often best represented by differential equations, while the results from ML models could not explain the mechanistic models in essence and extrapolation would be constrained within limited time under accuracy permission[63]. Hence, these above defects suggest that ML models could not completely replace differential equation models utilized in diverse individual circumstances. There is a desired goal to reduce the cost of data collection from nonlinear systems and computational resources in numerical experiments, which inspired academic research on exploring universal method combining with existed computational techniques. Recent study has investigated how to merge differential equation models and the desired advantages of ML models in some cases by different data-driven methods[47, 48, 53].

Subsequently, physics-informed neural networks (PINNs) were proposed by Raissi et al.[52, 54], in order to solve nonlinear problems through applying artificial neural networks for learning

underlying mathematical physics models. Specifically, this special methodology was design to encode the physical laws and obtain the corresponding mathematical model by handling some learning tasks from general nonlinear partial differential equations. The advance on PINNs made extraordinary progress as an academic data-driven discovery method in the fields of ML, which aroused great attention to study ML methods for identification of underlying mathematical functional structure .

Meanwhile, direct scientific interpretability of dynamical systems was always desired by scientists and even industry workers, for fully understanding and exploring underlying differentiable equation-oriented models. Concretely, during the engineering design process, series of dynamical models and functional processes are created to meet some specific practical satisfaction and optimally reach some target criterion according to complex mechanical systems[12]. For example, while designing an aerofoil, various factors including aerodynamic forces and geometry of aerofoil should be considered through the design stage to serve different flight regimes[62]. Therefore, the related ML techniques required to augment the computational efficiency and accuracy of training experimental data and further extrapolating beyond the sample districts, since finding the corresponding overall physical model allows researchers to simulate the most of important dynamical behaviours in those complicated integrated dynamics systems.

In present study, considering domain knowledge based on partially known models has gained increasing popularity in the fields of numerical analysis, in which the goal is to reconstruct the potential scientific models from sample data by ML techniques. The learnt models from collected data could be represented by symbolic differential equations and then structure their governing mathematical models. Inspired by the mixture of efficient data-driven ML techniques, such as symbolic evolutionary methods[13] and regressions[50], Rackauckas et al.[51] proposed how to combine ML with Universal Differential Equations to scientifically-based learn initially missing terms or unknown governing equations of machine-learnable models[63].

1.2 Problem Statement

In recent research, universal differential approximation has become a popular method according to ML techniques, where the appearance of universal differential equations model[51] aroused great attention in the fields of numerical analysis. To examine whether the numerical method is available to study nonlinear dynamics and discover the fundamental model or not, we will use a existed dataset recorded from a specific nonlinear dynamical system to put this idea into practice.

Moreover, there are lots of kinds of nonlinear dynamics system. In our work, we focus on studying Duffing-like model based on Duffing differential equation. Duffing-like differential equations generally consist of polynomial and periodic terms, which reveals some specific dynamical characteristics of a nonlinear system. To promise the accuracy and efficiency of universal differential equations model, Duffing model should be tested at the initial step. Then, we will

utilize the method to learn potential dynamical relationships from a series of existed datasets collected by a special experimental rig, which is introduced in Section 4.2.1

1.3 Stakeholder Relevance

With the development of numerous engineered products, there exists the dilemma that the superposition of potential theoretical models from their design process could capture the mainly important behaviours of those products but not exactly match the actual situations since the complicated damping mechanisms[63]. Aiming to construct more accurate representative models by computational simulation and algorithmic prediction, the dynamical system discovery problem has aroused excellent attention in applications to a wide variety of fields. There are several inspiring examples, consisting of saddle-node bifurcations with noninvasive control[6], simulation of Floquet multipliers[15] and aero-elastic flutter bifurcations[31]. The goal of this project is to explore how to apply UDE models in these dynamical systems, specifically the Duffing-like system described in the paper[7], for identifying their underlying physical model and correcting them from their mechanically experimental data respectively, with the 'greedy' desire for more accuracy than previously existed numerical experiments.

In the history of development on mathematical methodology and numerical technology, ML has been thought of as a key role in the field of nonlinear dynamics, especially considering their sophisticated-detected behaviours. However, there exists huge demand of more accurate and efficient algorithmic methods to satisfy the big data from various industries, which is eager to find the corresponding knowledge-enhanced model. Meanwhile, since the research on nonlinear dynamical systems refers to a wide variety of fields including biology, chemistry, engineering, economics, medicine and even history, mature excellent numerical methods have large market for researching. Hence, as a new popular method which has great performance on learning data, the universal differential equations model is able to benefit lots of stakeholders from related industries which need ML technique for predicting system's future variation tendency.

1.4 Outline

The structure of this paper is as follows. Firstly, in the Chapter 2, we review the previous literature on the development from artificial neural networks and briefly provide the mathematical theory of how to train neural networks. Secondly, in the Chapter 3, we introduce theoretical knowledge on universal approximation, consisting of universal approximation theorem, three mainly important mathematical methods and the computing software Julia. Then, in the Chapter 4, preliminary experiment on duffing equation and numerical experiments are provided to test the efficiency and accuracy of ML-embedded universal differential equations in Julia. Eventually, the summary of this project and its future work can be found in the Chapter 5.

CHAPTER 2

ARTIFICIAL NEURAL NETWORKS

In this chapter, the literature review and mathematical description of Artificial Neural Networks (ANN) will be presented respectively. In Section 2.1, we will introduce the historic research and development on machine learning techniques about ANN in details. In Section 2.2, we will provide the basic theoretical knowledge of general kind of artificial neural networks.

2.1 Literature Review on ANN

Inspired by neurophysiological knowledge, Neural Networks were originated by McCulloch and Pitts in 1943[39] with the purpose of finding numerical representations of biological information processing by building a mathematical model. After several years, based on the learning assumption by mechanism of neural plasticity, Hebbian learning was proposed by Hebb[25] and utilized to realize functions in computer by Farley and Clark[21]. In the mid-1960s, multi-layer functional networks was firstly used for data handling by Ivakhnenko and Lapa[29]. After that, backpropagation was proposed by Kelley[30] and Bryson[11] based on dynamic programming, which is one of the most important methods in ANN. Later, the technique of backpropagation was improved by Linnainmaa[34] that could be applied to train neural-network-like network.

Moreover, Speelpenning[58] designed an algorithm which is capable to automatically implement backpropagation intp differentiable models, and late Rumelhart et al.[57] shown the appearance of internal representations in the hidden layers. In the mid-1980s, a theory was published by Nielsen and Hornik et al.[28, 46] to indicate that training data by sufficient hidden units could approximate any continuous equations. Numerous advanced improvements of ANN could be also found that Ballard[5] published autoencoder hierarchies in feed-forward neural networks, based on unsupervised learning in 1987, and Hochreiter and Schmidhuber[27] presented supervised long short-term memory in recurrent neural networks in 1997. Afterwards,

the emergency of Bayesian neural networks has aroused great attention and develop artificial neural networks for classification and regression applications[45].

Motivated by constructing machine learning models based on prior domain knowledge[36, 48, 52], Raissi et al. proposed Physics-Informed Neural Networks (PINNs)[54]. PINNs was designed to combine ANN techniques with underlying some given physical relationships described by time-dependent nonlinear differential equations[54]. As an emerging universal function approximator, it was considered as a special data-driven approach of neural networks in supervised learning and systems identification with computationally satisfactory accuracy and efficiency, rather than other numerical methods. However, the final results of PINNs, as well as other methods in ANN, do not demonstrate or explain any dynamical laws in physical worlds and do not indeed solve the differential equations. Likewise, the cost of a large amount of data to promise a high level of accuracy is prohibitive in many cases for nonlinear dynamics, and we desire that the final results can interpret the essential dynamical models, specifically in mathematical equations, so that we can precisely predict future behaviours, which also utilize the originally known mechanisms. Considering these drawbacks above, researchers started to make an attempt on modifying ANN to train given data set, in order to promise sufficient precision on approximation and reveal the potential govern mathematical models.

With the development of artificial neural networks and the massive demand for industrial data processing of nonlinear dynamical systems, neural network models have been considered as an efficient method due to their surprising ability in learning complicated data set[4, 43, 64]. Bongard and Lipson[3] made a significant breakthrough on a new approach to generate symbolic equations from time series data in a nonlinear dynamical system, which encouraged the researchers to associate sparse regression with machine learning on system identification problems to discover the underlying physical models[10, 38].

2.2 Mathematical Description

In recent study in the fields of numerical analysis, Artificial Neural Networks have been widely used in parametric adjustment of dynamical system models. It has been shown that ANN is a data-efficient ML technique in some scientifically industrial problems and enables the computer to accurately simulate nonlinear dynamics and learn governing physical structure by training given data[4].

An artificial neural network is basically a series of functional transformations to train given data, which is realized by numeircal algorithms in computer. At the beginning step, input a set of data values from an external sample, denoted by $\mathbf{X} = [x_1, x_2, \cdots, x_k]^T$. Secondly, we construct linear combination in the first layer of the network by weights $\mathbf{W}_1 \in \mathbb{R}^{M\times k}$ and biases $\mathbf{b}_1 \in \mathbb{R}^{M\times 1}$

$$\mathbf{Y}_1 = \mathbf{W}_1\mathbf{X} + \mathbf{b}_1,$$

where each element y_i in $\mathbf{Y}_1$ is called activation, $i = 1,2,\ldots,M$ with respect to M artificial neurons in the first layer. Choose a suitable differentiable and nonlinear activation function that we denote $\sigma_1(\cdot)$ and transform $\mathbf{Y}_1$ through the given vector calulator

$$\sigma_1(\mathbf{Y}_1) = [\sigma_1(y_1), \sigma_1(y_2), \cdots, \sigma_1(y_M)]^T,$$

where each element in $\sigma_1(\mathbf{Y}_1)$ is called hidden unit. The typical choices of activation functions are nonlinear, which will be shown later. In order to acquire more accurate approximation and larger prediction horizon, deeper neural network should be required for training data. Hence, we train neural network by multiple hidden layers in general cases, and each layer in the neural network has the same linear and nonlinear transformation as above:

$$\sigma_j(\mathbf{Y}_j) = \sigma_j(\mathbf{W}_j\mathbf{Y}_{j-1} + \mathbf{b}_j),$$

where $\mathbf{W}_j \in \mathbb{R}^{N\times M}$, $\mathbf{b}_j \in \mathbb{R}^{N\times 1}$, $j = 2,3,\ldots,d-1$ for d hidden layer in this deep neural network. Ultimately, computer is able to output the accomplishment of some specific tasks, such as approximation or identification,

$$NN_\theta(x) = \mathbf{W}_d\sigma_{d-1}(\mathbf{W}_{d-1}\sigma_{d-2}(\cdots\sigma_1(\mathbf{W}_1\mathbf{X} + \mathbf{b}_1)\cdots) + \mathbf{b}_{d-1}) + \mathbf{b}_d,$$

where the parameter set is given by $\theta = \{\mathbf{W}_1,\ldots,\mathbf{W}_d;\mathbf{b}_1,\cdots,\mathbf{b}_d\}$, $\mathbf{W}_d \in \mathbb{R}^{k\times N}$, $\mathbf{b}_d \in \mathbb{R}^{k\times 1}$.

In addition, there are various activation functions which can be used into different cases to fit research models. We provide some typical activation functions as follows. All of the following activation functions are capable learn the nonlinearity between input and output from each layer. The difference amongst them is revealed by their corresponding mathematical properties, however, which does not decisively influence numerical performance in the most of cases.

- Identity Activation Function
$$I(x) = x$$
- Binary Step Activation Function
$$B(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geqslant 0 \end{cases}$$
- Sigmoid Activation Function
$$\sigma(x) = \frac{1}{1+e^{-x}}$$
- Hyperbolic Tangent Activation Function
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
- Softplus Activation Function
$$f(x) = \ln(1+e^x)$$

- Rectified Linear Unit (ReLU) Activation Function

$$\text{ReLU}(x) = \max\{0,\ x\}$$

- Gaussian Error Linear Unit (GELU) Activation Function

$$\frac{1}{2}x\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

- Exponential Linear Unit (ELU) Activation

$$\text{ELU}(x) = \begin{cases} \alpha\,(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

  with parameter α.

CHAPTER 3

UNIVERSAL APPROXIMATION

In this chapter, the development of universal differential equations and the process of learning data by training neural networks will be presented and explained in details, which is divided into four main parts. In Section 3.1, we present a popular numerical method called sparse identification of nonlinear dynamics (SInDy) to identify nonlinear dynamics by a sparse set of available mathematical equations. In Section 3.2, we introduce one of fundamental theories on the field of neural network, universal approximation theorem, which indicates that any continuous function can be approximated by sufficiently large but finite neural networks. In Section 3.3, we show universal differential equations (UDE) model which is a special machine learning technique based on scientific model. In Section 3.4, we introduce some optimization methods which can be utilized through training neural network in UDE model. In Section 3.5, we briefly outline the strong power of learning data and advantages of using UDE model by Julia.

3.1 Sparse Identification of Nonlinear Dynamics

Sparse identification of nonlinear dynamics is one of emerging methods for inferring nonlinear dynamics system, which arouses great attention recently from academics who do research on machine learning domain. Started from a paper published by Brunton et al.[10], SInDy was proposed to simulate nonlinear dynamics and discover the corresponding governing physical equations from observational data. Meanwhile, in the same year, Mangan et al.[37] applied implicit-SInDy method to successfully infer the structure and dynamics of biological networks. A later attempt of using SInDy can be found in the paper by Kaiser et al. in 2018, which presents that the application of SInDy is capable to enhance the performance of model predictive control even though based on limited amount of noisy data. The research during recent years has given

the evidence of the outstanding performance of SInDy method on algorithmically identifying and evaluating nonlinear dynamical systems and discovering the approximated functionality, as well as guaranteeing excellent robustness and accuracy.

Now, start the explanation on mathematical calculation of SInDy. At first, fix a dynamical system described in the following term by its corresponding target function f:

$$\dot{\mathbf{x}} = f(\mathbf{x}(t)), \tag{3.1}$$

where function f generally contains a few terms. The next step is to collect actual data from the dynamical system by substantial experiments. Assume that it is capable to learn the governing mathematical equations of the dynamical system from available time-series data $\mathbf{X}$ by the library consisting of finite number of simple functions. Denote the data matrix in the form of

$$\mathbf{X} = [\mathbf{x}(t_1), \mathbf{x}(t_2), \cdots, \mathbf{x}(t_k)]^T, \tag{3.2}$$

where t_1, t_2, ..., t_k are time samples, and the vector $\mathbf{x}(t_i)$ $(i = 1, 2, \ldots, k)$ is given by

$$\mathbf{x}(t_i) = [x_1(t_i), x_2(t_i), \cdots, x_m(t_i)]$$

And its corresponding derivative matrix has the following form

$$\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1), \dot{\mathbf{x}}(t_2), \cdots, \dot{\mathbf{x}}(t_k)]^T, \tag{3.3}$$

where the vector $\dot{\mathbf{x}}(t_i)$ $(i = 1, 2, \ldots, k)$ is given by

$$\dot{\mathbf{x}}(t_i) = [\dot{x}_1(t_i), \dot{x}_2(t_i), \cdots, \dot{x}_m(t_i)].$$

For the derivative $\dot{\mathbf{X}}$ which is realistically difficult to collect from actual experiment, we can utilize interpolation method to derive it approximately from given data matrix $\mathbf{X}$. Specifically, we use backward difference formulate in the later numerical experiments.

Notice that SInDy algorithm requires the dynamics are sparse in the chosen basis and it is far likely to fail on learning data if the dynamical system is not sufficiently sparse.

The next step is to select suitable candidate functions for algorithmic experiments. The criteria of selection on basis chiefly is in accord with the mathematical characteristics and dynamical properties of the training system, which can be observed from its experimental result or analysed from its physical features. Here is a motivating example of a library on candidate function space which includes up to n degree polynomial and trigonometric functions:

$$\Theta(\mathbf{X}) = \left[1, \mathbf{X}, \mathbf{X}^2, \ldots, \mathbf{X}^n, \sin(\mathbf{X}), \cos(\mathbf{X})\right], \tag{3.4}$$

where $\mathbf{X}^2$ denotes the quadratic nonlinearities in the state matrix with the following form:

$$\mathbf{X}^2 = \begin{pmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \cdots & x_2^2(t_1) & \cdots & x_m^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \cdots & x_2^2(t_2) & \cdots & x_m^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1^2(t_k) & x_1(t_k)x_2(t_k) & \cdots & x_2^2(t_k) & \cdots & x_m^2(t_k) \end{pmatrix}$$

In this example, the functional library matrix $\Theta(\mathbf{X})$ has dimensions $k \times (n+3)$, where it requires $k \gg n+3$ since the value of k represents the number of collected data samples and it should sufficiently large to guarantee the identification accuracy. Thus, in order to deal with the dynamical system numerically, the corresponding sparse regression problem can be set up, under the constraint that only a few nonlinearity terms are active :

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi, \tag{3.5}$$

which is determined by the sparse basis $\Xi = [\xi_1, \xi_2, \cdots, \xi_m]$ with coefficients of each column ξ_j. Identify the active terms amounts to solve the regression problem by minimizing the following objective function to find the sparse basis Ξ :

$$\Xi = \arg\min \left\{ \| \dot{\mathbf{X}} - \Theta(\mathbf{X})\Xi \|_2 + \lambda \| \Xi \|_1 \right\}. \tag{3.6}$$

In addition, there unavoidably exists noise within a dynamical system in general scenes, which may contaminate both of data matrix $\mathbf{X}$ and $\dot{\mathbf{X}}$. Hence, we can also consider the other regression formula in data analysis instead[10]:

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi + \eta\mathbf{Z}, \tag{3.7}$$

where $\mathbf{Z}$ denotes the matrix consisting of independent identically distributed Gaussian entries with zero mean, and η is the noise magnitude.

To sum up, it can be seen that SInDy is able to reconstruct the dynamical system by symbolic equations through training given data without any domain knowledge, which solves the problems of interpretability of mechanistic models that ANN could not achieve as above description[63]. However, the present existed techniques of SInDy could not extract information according to previously known physical knowledge, which may limit its accurate and efficient approximation ability in some cases, especially when the previously known mathematical models can capture the most of mainly important characteristics of the dynamical system. Moreover, there still exists potential conflict on sparse representation, which is considered the most challengeable problem in the application of SInDy[63]. Therefore, collectively, these studies outline a critical role for SInDy on technical combination of dynamical system and machine learning, which is a newly-developing but feasible and valuable research direction.

3.2 Universal Approximation Theorem

The first serious discussion and analysis of universal approximation emerged during the late 1980s with the paper published by Hornik et al.[28]. In their article, they proposed that multilayer feedforward networks can be considered as universal approximators to estimate any Borel measurable function on finite space with any required accuracy. After that, research investigating the mathematical theory associated with training neural networks has paid attention on the

approximation ability of different kinds of ANN on Euclidean spaces. There are two main categories of universal approximation theorems called arbitrary width case and arbitrary depth case respectively.

Research into the arbitrary width case has more than 30-year history. In 1989, Cybenko[14] firstly proved the approximation capability of learning nonlinearities by using sigmoid activation function in ANN. During the 1990s, the theorem was further studied by Leshno et al.[32] and then Pinkus[49] that nonpolynomial activation function has the potential of universal approximation. There are several other version of arbitrary width case which can be found in [23, 24], and the classical one is stated as follows:

Theorem 3.1 (Universal Approximation Theorem (Arbitrary Width Case))**.** *For a given activation function $\sigma: \mathbb{R} \to \mathbb{R}$ and positive integers d, D, function σ is not a polynomial function if and only if, for every continuous function $f: \mathbb{R} \to \mathbb{R}$, every compact subset K of $\mathbb{R}^d$, and every $\epsilon > 0$ there exists a continuous function $f_\epsilon: \mathbb{R}^d \to \mathbb{R}^D$ with the following form:*

$$f_\epsilon = W_2 \circ \sigma \circ W_1,$$

where W_1 and W_2 are composable affine maps, and $\circ$ denotes component-wise composition, such that the approximation bound

$$\sup_{x \in K} \| f(x) - f_\epsilon(x) \| < \epsilon$$

holds for any arbitrarily small ϵ.

The above theorem provides basic theoretical knowledge that the output function f_ϵ from a single hidden layer of neural network is capable to approximate any smooth target function f by an arbitrary number of artificial neurons.

Turning now to the arbitrary depth case as the depth goes to infinity. It is an extension on diverse perspective through the infinite depth and bounded width. In this case, there are an arbitrary number of hidden layers with only one neuron for per layer. In most recent studies, the transformation of universal approximation theorem was well established by some scientists. In the paper by Lu et al. in 2017[35], the result provides the evidence that width-$n+4$ Rectified Linear Unit (ReLU) deep networks are capable to act as universal approximators, where n is the input dimension, and depth is likely to have more effectiveness than width in this type of network. The similar exploration on ReLU networks can be also found in the paper in 2018 by Lin and Jegelka[33]. The universal approximation theorem for the arbitrary depth case can be stated as follows:

Theorem 3.2 (Universal Approximation Theorem (Arbitrary Depth Case))**.** *For any Bochner-Lebesgue p-integrable function $f: \mathbb{R}^n \to \mathbb{R}^m$ and any $\epsilon < 0$, there exists a fully-connected ReLU network F of width exactly $d_m = \max\{n+1, m\}$, such that the integral approximation bound*

$$\int_{\mathbb{R}^n} \| f(x) - F(x) \|^p \, dx < \epsilon$$

holds for any arbitrarily small ϵ.

Furthermore, there exists a function $f \in L^p(\mathbb{R}^n, \mathbb{R}^m)$ and some $\epsilon > 0$, for which there is no fully-connected ReLU network of width less than $d_m = \max\{n+1, m\}$ satisfying the above approximation bound.

3.3 Universal Differential Equations

In this section, we introduce universal differential equations model, which is an innovative and profitable method by combining ANN and ordinary differential model. Based on scientific structures, UDE model improves neural networks technique and benefits the machine-learnable models. The Universal Approximation Theorem we discuss in Section 3.2 demonstrates that, sufficient large neural networks with nonlinear activation function can approximate any continuous function on compact subsets of Euclidian space. According to this significant theorem and considerable techniques of ANN, UDE model was proposed firstly by Rackauckas et al.[51] last year, which is regarded as a universal approximator embedded in a portion of differential equations. It aims to extend data-driven models to incorporate mechanistic features and potential physical laws for practical engineering problems that there is a high demand to learn the governing scientific model based on existed datasets.

For a given differential equation model, there are some missing terms which need to be found through training this corresponding UDE. Define the missing terms as universal approximator $U_\theta(t)$, then find it by minimizing the following cost function for the given data set $\mathbf{X} = \{x_i\}_{i=1}^k$ with $x_i = x(t_i)$

$$U_\theta(t) = \arg\min \left\{ \sum_{i=1}^{k} \| U_\theta(t_i) - x(t_i) \| \right\}.$$

Considering gradient-based optimization methods to train distinct types of differential equations, this problem becomes calculating gradients $\frac{dU}{d\theta}$ with respect to parameters. A special differentiable programming framework has been constructed for reverse-mode accumulation in [51]. Define the pullback at x for a given function $x = f(t)$ as

$$B_f^t(x) = x^T J,$$

where J is the Jacobian matrix of this function. In the computational programming, the function f can be disassembled into L discrete processes as

$$f = f^L \circ f^{L-1} \circ \cdots \circ f^1.$$

Then the multiplying Jacobian matrix for a vector v can be computed by

$$v^T J = (\cdots((v^T J_L) J_{L-1}) \cdots) J_1,$$

which can be implemented by embedded deep neural networks. Hence, the reverse decomposition of the pullback can be discretized by

$$B^t_f(\mathbf{X}) = B^t_{f^1}\left(\cdots\left(B^{t_{L-2}}_{f^{L-1}}\left(B^{t_{L-1}}_{f^L}(\mathbf{X})\right)\right)\cdots\right),$$

where $t_i = (f^i \circ f^{i-1} \circ \cdots \circ f^1)(t)$. After training UDE and find the optimized parameters, we apply the SInDy algorithm to find the governing equations which can perform according with the trained neural network.

The numerical discovery approach was chosen because it has a number of attractive features. UDE models allow computer to consider previously known domain knowledge described by differential equations, which can constrain the problem in order to reconstruct more data-efficient learnable dynamical systems. It is great compatibility of ANN and SInDy, which augments existing symbolic mathematical rules and also guarantees excellent computational efficiency since it just requires a short time series of data for accurate extrapolation[63]. There is the open-source ML-embedded UDE programming of DiffEqFlux.jl[51] based on Julia which will be introduced in Section 3.5.

In this article, we focus on universal ordinary differential equation (UODE) model to serve the later discussion of Duffing-like system. For a fixed dynamical system described by universal ordinary equations , there is a specific case that scientists have already known parts of governing mathematical model and collected a short time series data from the system. Then, for the ordinary differential equation as Equation (3.1), we can construct knowledge-based UODE to learn the residual unknown mathematical interactions from the given information as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t)) + U_\theta(\mathbf{x}), \tag{3.8}$$

where the state variable $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ depends on time t, and the system has n differential equations given by $\mathbf{f} = (f_1, f_2, \ldots, f_n(t)) : \mathbb{R}^n \to \mathbb{R}^n$. During the functional approximation process, the universal approximator $U_\theta : \mathbb{R}^n \to \mathbb{R}^n$ has n dimensions, which is utilized to train acted as a neural network. Moreover, considering a more complex case with the other type of target function $f(t, \mathbf{x}, \dot{\mathbf{x}})$, we can exploit the corresponding sophisticated universal approximator as shown in the following UODE:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \dot{\mathbf{x}}) + \widetilde{U}_\theta(t, \mathbf{x}, \dot{\mathbf{x}}). \tag{3.9}$$

Besides, there are also some other classifications of differential equation models which can utilize UDE technique as shown in the following brief presentation.

- Stochastic Differential Equations (SDE)

  $$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t,$$

  where W_t is the Brownian motion.

- Delay Differential Equations (DDE)

$$\frac{dx(t)}{dt} = f(t, x(t), x_t),$$

where $x_t = \{x(\tau) : \tau \leqslant t\}$ represents the trajectory of the state variable x in the past.

- Differential-Algebraic Equations(DAE)

$$\mathbf{F}(\dot{\mathbf{x}}(t), \mathbf{x}(t), t) = 0,$$

where $\mathbf{x} : \mathbb{R}^2 \to \mathbb{R}^n$ is a vector of dependent variables $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]$ and the vector function has the map $\mathbf{F} = (F_1, F_2, \ldots, F_n(t)) : \mathbb{R}^{2n+1} \to \mathbb{R}^n$.

- Partial Differential Equations (PDE)

$$F\left(t, x, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial x}, \nabla u(t,x)\right) = 0$$

3.4 Derivative-based Methods

Derivative-based optimization plays a significantly important role in many fields of science and engineering. It is a first-order iterative optimization algorithm to approach a local minimum (or maximum) of a specific differentiable function. In our work, the differentiable function is generally the loss function for minimizing L2-error, which is given by the following form:

$$\mathscr{L} = \sum_{i=1}^{N} (U_\theta(t_i) - x_i)^2, \tag{3.10}$$

where U_θ is the universal approximator and the N data points (t_i, x_i) for $i = 1, 2, \ldots, N$ belong to the collected time series datasets. The basic idea of derivative-based method is to compute repeated steps in the opposite (or synclastic) direction of the approximate gradient of the function at the current point for a fixed number of iterations which is based on the required accuracy and efficiency.

According to different efficient local derivative-based methods, some complicated practical problems can be numerically solved by optimization of parameterized objective function. There are several traditional derivative-based optimization methods such as Steepest Descent Method[17], Conjugate Gradient Method[26] and Newton–Raphson Method, which have been typically used in the past to investigate the differential properties and optimize the parameter values. The general algorithmic process of gradient-based optimization for a given continuous and scalar objective function $f(x)$ can be illustrated as follows:

1. Examine the convergence: Start with integer $i = 0$. Continue the numerical iterations given by x_i until the fixed optimality condition for convergence is satisfied by solution x_N. Then x_N is the final result solution.

2. Compute the direction of gradient: Calculate the directional vector d_i. The corresponding computing formula is determined by the classification of the chosen optimization method.

3. Update the variables x_{i+1}: Set $x_{i+1} = x_i + \alpha_i d_i$, where the value of the positive scalar α is obtain by comparing $f(x_{i+1})$ with $f(x_i)$.

In this work, we will utilize two recently popular derivative-based methods which currently exist for the measurement of parameter values in nonlinear dynamical system. The first one is Adaptive Moment Estimation (ADAM) proposed by Kingma and Ba in 2015[18]. It is an efficient stochastic optimization method derived from adaptive moment approximation by computing rates for each parameter, which combines the advantages of AdaGrad[20] for sparse gradients and RMSProp[60] for on-line and non-stationary settings. Fix a noisy scalar objective function $f(\theta)$ with respect to parameters θ. The convergence condition is designed by

$$R(N) = \sum_{i=0}^{N} [f_t(\theta_t) - f_t(\theta^*)] \leqslant \epsilon_R, \tag{3.11}$$

with optimal parameters $\theta^* = \arg\min \sum_{i=0}^{N} f_t(\theta)$ and convergence parameter ϵ_R. In each iteration of the algorithm, the gradients m_i and squared gradients v_i are updated by the following formulas:

$$m_{i+1} = \beta_1 m_i + (1-\beta_1)\nabla_\theta f_t(\theta), \tag{3.12}$$

$$v_{i+1} = \beta_2 v_i + (1-\beta_2)(\nabla_\theta f_t(\theta))^2, \tag{3.13}$$

where β_1, $\beta_2 \in [0,1)$ are hyper-parameters. And the parameters θ are adapted by ADAM update rule:

$$\theta_{i+1} = \theta_i - \frac{\alpha_i m_i}{(1-\beta_1^i)\left(\sqrt{\frac{v_i}{1-\beta_2^i}} + \epsilon\right)}, \tag{3.14}$$

where ϵ represents an arbitrarily small quantity. In addition, ADAM is able to keep an exponentially decreasing average for both of the gradients and the squared gradients.

The second numerical method is Broyden–Fletcher–Goldfarb–Shanno (BFGS) Method, proposed by Fletcherin 1989[22] and improved by Mogensen and Riseth in 2018[42] into using Julia. Superseding Davidon–Fletcher–Powell (DFP) Method[16], BFGS is able to determine the descent direction through preconditioning the gradient by curvature information. Based on the sequence of function values, the optimality condition of BFGS is given by

$$R(N) = \| \nabla f(x_N) \| \leqslant \epsilon_R. \tag{3.15}$$

In each iteration, the update rule for directional vector d_i, solution x_{i+1} and adaptive matrix V_i which is the inverse of a symmetric positive definite matrix can be found in the following

equations:

$$d_{i+1} = -V_i \nabla f(x_i), \tag{3.16}$$

$$x_{i+1} = x_i + \alpha_i d_i, \tag{3.17}$$

$$V_{i+1} = \left[I - \frac{q_i p_i^T}{q_i^T p_i} \right] V_i \left[I - \frac{p_i q_i^T}{q_i^T p_i} \right] + \frac{q_i q_i^T}{q_i^T p_i}, \tag{3.18}$$

with respect to $q_i = \alpha_i d_i$ and $p_i = \nabla f(x_{i+1}) - \nabla f(x_i)$.

3.5 Julia SciML

Julia is one of practical and effective programming languages for studying dynamical systems. It gives extraordinary performances by providing interactive use and dynamically typed scripting. Recently, base on Julia language, Scientific Machine Learning Software (SciML)[3] was created to support modular scientific simulation and unify the packages for scientific machine learning, which is a popular open-source software organization. Most of numerical techniques in Julia SciML were based on diverse differential equations, which is able to achieve accurate and efficient exploitation in reproducible environments and express many object-oriented and functional programming patterns[2]. All the work on the computer was carried out using Julia in this article.

Universal differential equations model was proposed to combine the machine learning techniques with mathematical methodologies of differentiable models in 2020[51], which enriched the Julia library sources on many aspects. The open-source codes are developed to automatically utilize universal differential equation solvers for different classes of differential equations, including PDE, SDE and DAE as shown in Table 3.1, which summarizes some popular open-source machine learning libraries in Julia software from this online organization. Meanwhile, abased on the abundant machine learning technologies, the algorithmic sources also benefit the research on nonlinear dynamics.

Packages	Functionality
DifferentialEquations.jl	Differential equations solvers
DiffEqBayes.jl	Bayesian estimation for differential equation models
DiffEqFlux.jl	Universal differential equation models
DataDrivenDiffEq.jl	Estimate Koopman operators
ReservoirComputing.jl	Echo State Networks
SparsityDetection.jl	Detect the sparsity patterns of Jacobians and Hessians
SparseDiffTools.jl	Combine DifferentialEquations.jl with DiffEqFlux.jl
NonlinaerSolve.jl	Nonlinear solving packages by using Jacobian construction
DiffEqParamEstim.jl	Estimate parameters

TABLE 3.1. Important open-source tools for scientific machine learning in Julia

CHAPTER 4

NUMERICAL EXPERIMENT

In this chapter, the whole process of correlate numerical experiment and further mathematical description will be exhibited, which is divided into two main parts to present Duffing system in Section 4.1 and Duffing-like system in Section 4.2 respectively. Moreover, there are three separate portions within Section 4.2, in order to discuss the effect from additional noise on learning a specific dataset.

4.1 Duffing System

In order to guarantee the high accuracy on predicting Duffing-like system, the initial step is to determine the precision on predicting Duffing model by using numerical simulation according to UDE model. Hence, this section is to show the preliminary experimental result about testing UDE on Duffing system. The following Duffing equation is regarded as our motivating example:

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma\cos(\omega t), \tag{4.1}$$

where $\delta, \alpha, \beta, \gamma, \omega$ are constant parameters.

Now, the goal is to test the third-oder term βx^3 by universal approximator $U : \mathbb{R} \to \mathbb{R}$, which would be utilized as a neural network to learn the assuming missing term which is incorporated in this system of ordinary differential equations. Denote $x_1 = x$ and $x_2 = \dot{x}$, then we can have the corresponding knowledge-based UODE with the form:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \delta x_2 - \alpha x_1 + U(x_1, x_2) + \gamma\cos(\omega t) \end{cases} \tag{4.2}$$

Set values of constant parameters for the Duffing model by $\alpha = 1, \beta = 3, \gamma = 2, \delta = 1, \omega = 2$. Considering that there exists small noise within experimental data in general cases, we add

noise with magnitude 0.01 into ideal data simulated by Equation (4.4) for $t \in [0,3]$, in terms of mean value of ideal data[63]. Then, train neural networks with different kinds of activation functions through 3 hidden layers and 5 neurons in each layer, and utilize ADAM and BFGS as gradient-based optimization methods while training UDE. In this part, we compare three regular activation functions including Sigmoid function, Hyperbolic Tangent function and Softplus function.

During the numerical experiment, we train the universal approximator $U(x_1,x_2)$ to reconstruct the missing dynamical equation. Based on SInDy algorithm with a specific library of basis functions, the missing term can be detected into algebraic form. Considering the characteristics of Duffing system and even Duffing-like system, there are generally described by differential equation with polynomial and periodic functions. Therefore, we design the library of basis functions consisting of polynomial basis (up to degree 5), sine and cosine functions in our experiments. Hence, we have the identification $U(x_1,x_2) = -2.9677094x^3$. Compared with the true term $-3x^3$, the result is close to the exact original equations. The comparison among further extrapolation results can be found in Appendix (see Figure A.1). In order to select the best activation function to construct UDE model in the following other experiments, we compare the experimental error through these three activation function, and the training results can be found in Figure 4.1. It can be seen from the comparison of the error function among above three cases that the fluctuation tendency of error from learning Duffing model by UDE are similar. Thereinto, the error from Softplus activation function is more stable and smooth, and more likely to keep high precision and reduce prediction error at some intervals. Hence, Softplus function is a better choice as activation, rather than other two kinds of activation functions, for this kind of UODE problems in the following numerical experiments.

Now, consider the other case: assume that we miss the last term $-\beta x^3$ of LHS and the periodic term $\gamma\cos(\omega t)$ of RHS in Equation (4.1). Then, the corresponding knowledge-based UODE is given by

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_1 = \delta x_2 - \alpha x_1 + \widetilde{U}(x_1,x_2,t) \end{cases} \tag{4.3}$$

where $\widetilde{U}(x_1,x_2,t)$ is the universal approximator in this UODE problem. It is a more difficult and complex problem, since one of the missing terms relates to time and we need to try to consider adding time variable into predictor to train the network and restore the Duffing equation to the greatest extent. During this numerical experiment, we adding $\cos(\omega t)$ into the original set of basis functions. At the same way, we get the estimated result and give the universal approximator by SInDy algorithm in the form as

$$\widetilde{U}(x_1,x_2,t) = 0.20534347\dot{x} + 1.2271634\cos(\dot{x}) - 0.73734707x^2 - 0.4473314x^4 - 0.5589447x^5$$

Contract with the true term $-3x^3 + 2\cos(2t)$, this result is not accurate and acceptable (see Figure 4.2. The approximator losses the oscillator term $\gamma\cos(\omega t)$, and also has other uncorrelated

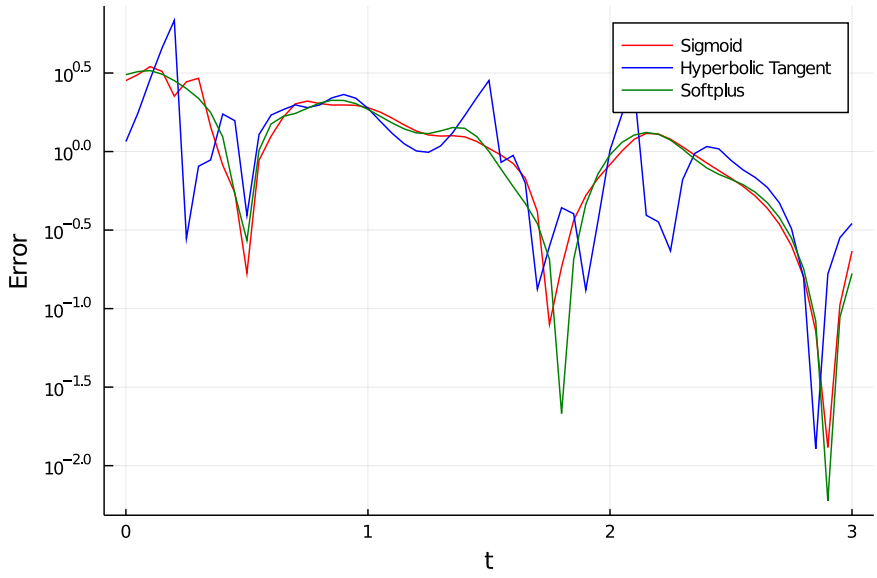


FIGURE 4.1. Compare experimental error among different activation functions including Sigmoid function in red, Hyperbolic Tangent function in blue and Softplus function in green.

and unexpected terms. The key problem is that the result directly ignore the important term $-3x^3$, which illustrates that considering adding time to learning Duffing-like dynamical system may be a bad direction due to its instability and inaccuracy. However, when we try to use the universal approximator $\widetilde{U}(x_1,x_2)$ with the former experiment, the estimated result is surprising

$$\widetilde{U}(x_1,x_2) = -2.995949x^3$$

Even though the simulated term losses the periodic term $2\cos(2t)$, it learns the term $-3x^3$ quite accurately, especially comparing with the universal approximator $\widetilde{U}(x_1,x_2,t)$. The approximated results on finding missing terms are shown in Figure 4.2. In this graph, the differences of the estimation deviation between the true solution and the other two universal approximators are not quite obvious within the training interval $t \in [0,3]$. Nevertheless, turning now to the experimental evidence on predicting the further dynamical behaviours, we can find that the difference between these two approximators.

The extrapolation result can be found in Figure 4.3, in order to distinguish the prediction ability between these two universal approximators. The comparison of the two extrapolation results obviously reveals that learning data in Duffing model without considering adding time as a part of universal approximator can acquire more accurate and appropriate prediction, where the universal approximator $\widetilde{U}(x_1,x_2)$ can catch the vital mathematical characteristic of the Duffing system in UDE model.

For the result from estimation by $U(x_1,x_2)$, we can find that the numerical fitting is capable

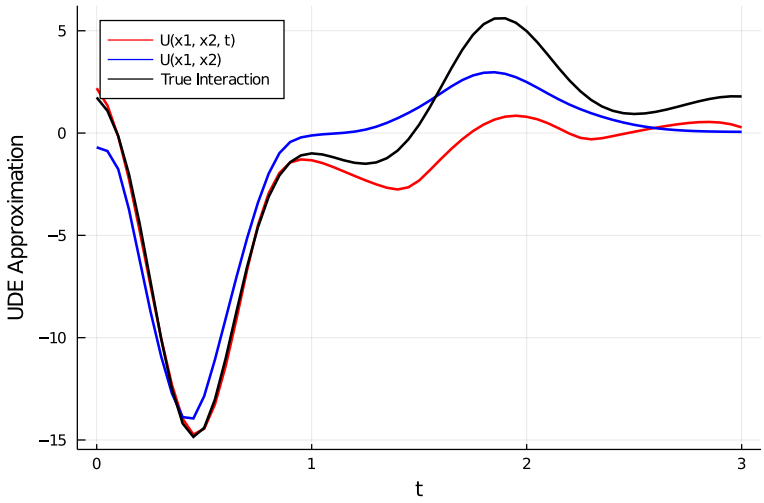


FIGURE 4.2. Estimate the missing term by two different universal approximators with $\widetilde{U}(x_1,x_2,t)$ in red and $\widetilde{U}(x_1,x_2)$ in blue.

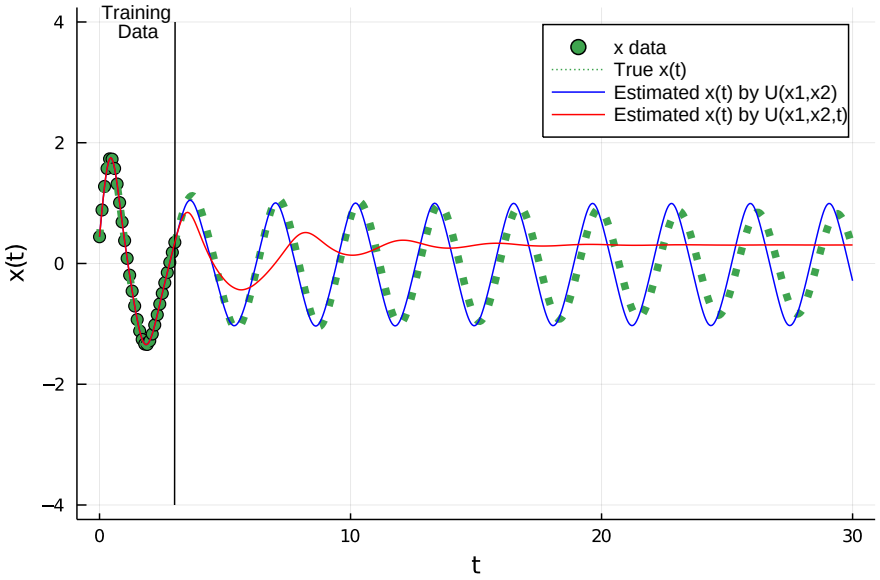


FIGURE 4.3. This figure is to compare extrapolation result by whether considering time: The green solid circles represent the original data within interval $t \in [0,3]$ which are utilized to extrapolate further prediction. The small green squares show the ideal solution within interval $t \in [0,30]$, which is simulated by original equations. The results through extrapolating data from $t \in [0,3]$ to $t \in [0,30]$ can be compared by two different universal approximator $U(x_1,x_2)$ in blue and $\widetilde{U}(x_1,x_2,t)$ in red respectively.

of extrapolating the further fluctuation from the short time series data and almost accurately

reconstruct the original Duffing model in the case of not discussing the periodic term, even though adding slight noise into simulated data, which implies the potent power of knowledge-enhanced approach.

These above results suggest that it is applicable to apply UDE to learn Duffing-like model. The next section, therefore, moves on to discuss the usage of UDE on learning nonlinear dynamics as the datasets shown in Section 4.2.1.

4.2 Duffing-like System

This section is divided into three parts. The first part introduces existing datasets from an academic paper published by Beregi et al. in 2020[7]. The second part shows the experimental results without the additional noise. And the third part presents the experimental results in the present of noise and compare the effect from different noise levels.

4.2.1 Datasets

Beregi et al. provide a comprehensive collection of datasets on a special device, which can be found within their paper "Robustness of nonlinear parameter identification in the presence of process noise using control-based continuation"[7]. In their experiment, the experimental rig is a nonlinear oscillator mounted on the shaker, and the corresponding data acquisition records the nonlinear behaviour and properties.

The datasets recorded abundant periodic orbits of the system. We focus on the fluctuation of the input voltage from the strain-gauge with respect to time at different level of noise for each branchpoint in our paper. In addition, each branchpoint dataset is collected with specific values of frequency and at a fixed noisy level, which also contains some applicable values of related parameters .

For each branchpoint from the experimental result, a linearly damped and Duffing-like oscillator of motion is described by the state variable x with respect to time t. Then, we have the differential model to simulate this nonlinear system in the form of the following equation:

$$\ddot{x} + b\dot{x} + x + \mu x^3 + \nu x^5 + \rho x^7 = \delta_{st}\cos(\omega t), \tag{4.4}$$

where b is the viscous damping, μ, ν, ρ are nonlinear coefficients, δ_{st} is the static deflection and ω is the forcing frequency. The values of the first four correlate parameters can be identified by control-based continuation at different noise levels. However, the value of the static deflection δ_{st} varies as the voltage variable x changes, which does not depend on time.

This Duffing-like model can mainly catch the dynamical characteristics and predict the nonlinear behaviours in general, and the parameter values can be identified from acquired datasets by Control-based Continuation accurately[7]. However, we are still curious about exploring more precise and underlying scientific model for further research, based on the Duffing-like model.

To summarize the above introduction on the datasets, there are three problems while learning the corresponding scientific model from the datasets by utilizing UDE. The first thing is to predict at noise-free datasets with constant frequency, which is the basic step to check whether the UDE model can identify this nonlinear dynamics with high precision. The second one is learning the experimental data by UDE in the case of varying frequency, which can decrease the negative limit from forcing frequency, compared by former study. The third difficulty is to consider the effect from the presence of noise in UDE model.

4.2.2 Without Additional Noise

At the beginning on learning the nonlinear system of the datasets introduced above, we discuss an ideal case that the additional noise dose not exist within the dynamics model, except some uncontrolled noise sources from experimental device. From the result of identification by using Control-based Continuation[7], we have the identified model parameter values $\bar{b} = 0.00798$, $\bar{\mu} = 0.2999$, $\bar{\nu} = -0.0258$ and $\bar{\rho} = -0.00025$, which will be utilized into the knowledge-based Duffing-like model in this section.

The experiment aims to observe the numerical result on learning the data with fixed frequency $\omega = 24$ Hz. As we examine above, the universal approximator $U(x_1, x_2)$ will be employed to train the neural network, where $x_1 = x$ and $x_2 = \dot{x}$. The corresponding knowledge-based UODE can be given by:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -bx_2 - x_1 - \mu x_1^3 - \nu x_1^5 - \rho x_1^7 + \delta_{st}\cos(\omega t) \end{cases} \tag{4.5}$$

Now, we choose the front part of a specific branchponit from the datasets within the first 207 data samples against internal time, the sequential order of which is based on the sample counter, consisting of the movement of the measured voltage from the strain gauge against time, i.e. the set $\left\{\left(t^{(i)}, x_1^{(i)}\right)\right\}_{i=1}^{N}$ with respect to $N = 207$ which is the number of collected samples considered in this experiment. Because the values of derivative $x_2 = \dot{x}$ are not involved in the datasets, we can apply backward difference formulate to access their approximate values as the following equation:

$$x_2^{(i+1)} \approx \frac{x_1^{(i+1)} - x_1^{(i)}}{t^{(i+1)} - t^{(i)}}. \tag{4.6}$$

Then, we train the data by neural network with Softplus activation function $f(x) = \ln(1 + e^x)$ through 3 hidden layers and 5 neurons in each layer, which is restricted by loss function to ensure that the learning process is approachable without unacceptable distortion of basic physically dynamical behaviours. As shown in Figure 4.4, the final training loss after 226 iterations can achieve convergence.

Learning by a set of basis functions consisting of up to degree 5 polynomial basis, sine and cosine functions, we have the universal approximator $U(x_1, x_2) = 0.34337473x + 0.18827073x^2$

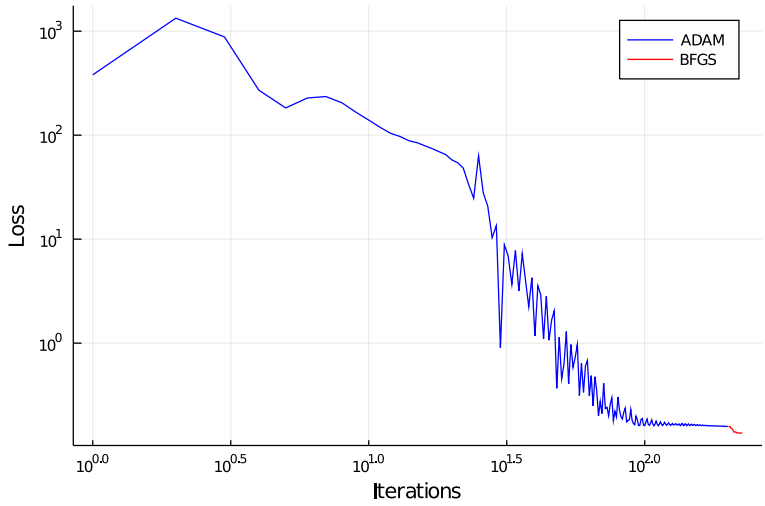


FIGURE 4.4. Loss function converges after 226 iterations on learning datasets at frequency $\omega = 24$ without additional noise by UDE model.

according to SInDy algorithm. Hence, the Duffing-like model is restored with the form as:

$$\ddot{x} + b\dot{x} + \alpha x + \beta x^2 + \mu x^3 + \nu x^5 + \rho x^7 = \delta_{st}\cos(\omega t), \tag{4.7}$$

where $\bar{\alpha} = 1.34337473$ and $\bar{\beta} = 0.18827073$.

Now, we extrapolate the constructed UDE model from the first 207 data samples to the first 832 data samples against internal time, and compare the result with the actual data, which can be found in Figure 4.5. The upper left graph provides the simulation of the universal approximator which contributes to the potential missing terms in the original differential model. The bottom graph shows the extrapolation result, which intuitively reveals that UDE model can precisely catch the nonlinear behaviours of the nonlinear system and accurately predict the future tendency in this specific case. Meanwhile, the upper right graph presents the absolute error of the extrapolation results by using UDE, compared by the actual datasets during these tests. The result numerically illustrates that the error can be controlled under 10^{-2}, which is significantly small and acceptable.

To sum up, UDE model can substantially learn the datasets without any additional noise besides intrinsic noise from experimental setup. Specifically, the extrapolation result inferred by UDE model is remarkably accurate, which reflects the strong and reliable simulation ability of UDE model for this Duffing-like system without considering the effect from supplementary noise.

4.2.3 Presence of Noise

In this section, we will discuss the ability of UDE model for learning the given datasets considering the presence of additional noise. In general experiments, the presence of noise is a more common

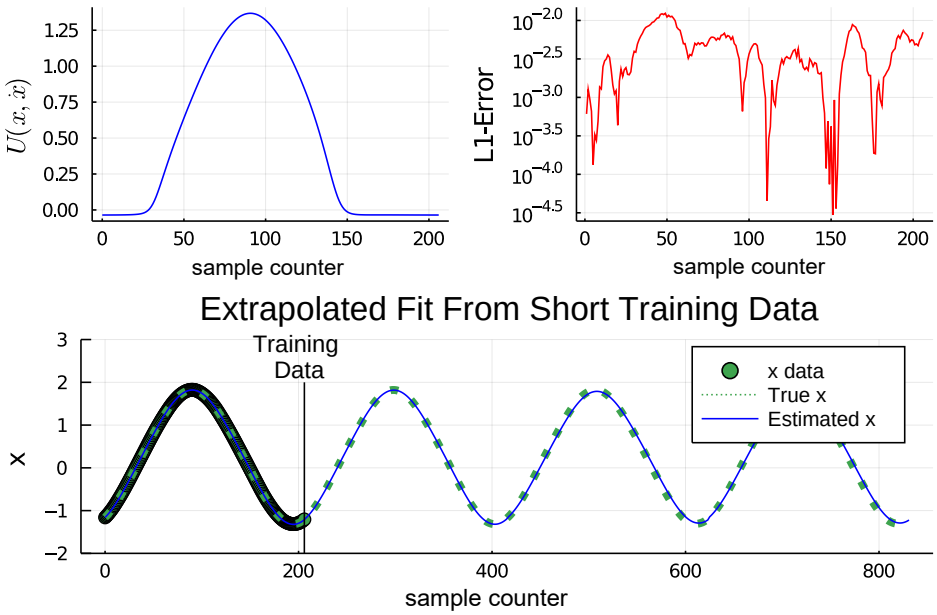


FIGURE 4.5. Extrapolation result without adittional noise: The upper left figure shows the universal approximator; the upper right figure present the experimental error; the bottom figure displays the extrapolated fit on blue curve from training the first 207 data samples to the first 832 data samples, compared with the actual datasets represented by green circles and further green small squares.

case rather than the ideal circumstance without any additional noise. The noisy data sometimes may bring unexpected influence to the experimental results. Hence, it is necessary to explore the effect from the presence of noise while testing an algorithmic method. Moreover, we hope the UDE model can also catch the key nonlinear dynamical characteristics and meet error tolerances, even though there exists non-negligible level of noise, which is able to eliminate the problems of collecting data contaminated by noise at the beginning process.

In the paper of Beregi et al.[7], the datasets are collected in a specific noise-polluted environment. They explore the dynamical results at different levels of noise and identify the model parameter values amongst these complex cases. We select three typical cases to discuss in our experiments, belonging to noise levels N0, N4 and N10. For these three particular noise level, the values of parameter b, μ, ν and ρ in the Duffing-like model are different identified approximatedly by Control-base Continuation. The corresponding identified model parameter values can be found in Table 4.1.

We employ the known identified parameters substituting into the Duffing-like model and find whether the UDE model can learn the data with frequency $\omega = 24$ or not. To simplify the experiments, we cut out a part of data within a whole period from N0, N4 and N10 separately, with respect to the first 208 data samples against internal time. The sequential order of is likewise

Noise	b	μ	ν	ρ
N0	0.00798	0.2999	-0.0258	-0.00025
N4	0.01217	0.3132	-0.0344	0.00114
N10	0.00578	0.3160	-0.0365	0.00130

TABLE 4.1. Parameter values identified by Control-based Continuation in the paper[7].

Noise Level	Universal Approximator $U(x,\dot{x})$
N0	$0.33052316x + 0.19383061x^2$
N4	$0.2690014 + 0.22576573x^2$
N10	$0.12953652\cos(x)$

TABLE 4.2. Universal approximator at different levels of noise computed by SInDy algorithm.

based on the sample counter given by the dataset $\left\{\left(t^{(i)}, x_1^{(i)}\right)\right\}_{i=1}^{N}$ with respect to $N = 208$. Then, train neural network by Softplus activation function $f(x) = \ln(1 + e^x)$ through 3 hidden layers with 5 neurons respectively based on the universal approximator $U(x,\dot{x})$, and then optimize the loss function by ADAM and BFGS. The loss function can converge after 226 iterations and 206 iterations in the cases of N0 and N4 respectively, however, fails to converge after 514 iterations when the noise level is N10. The result demonstrates that the optimization methods ADAM and BFGS is not applicable to control the loss when the additional noise is quite excessive in the UDE model.

The approximated simulation by learning data can be found in Figure 4.6 for these three cases. As can be seen from the result graphs, the UDE approximation curves almost overlap their corresponding actual data with noise level N0 and N4, which provides that the UDE learning technique is allowed to grasp the major dynamical performances and encounter the basic mathematical conducts, compared with their respective actual data at low level of noise. On the contrary, we is likely to acquire a negative approximation result in a higher level of noise, which can be fully explained as the result in the bottom figure for N10.

Meanwhile, according to SInDy algorithm, the estimated universal approximators for N0, N4 and N10 can be obtained as shown in Table 4.2, which can thoroughly conform with our expectation. When there is no additional noise in the nonliear system, the UDE model can detect the terms x and x^2. But when the additional noise level increases to N4, UDE technique is unable to find the term x and take the constant term instead of the first degree term. The most interesting aspect of the results is that there is a cosine term when comes to higher level of noise N10, which becomes completely different from N0 and N4. Compared by the case of lower noise, there is no polynomial term but a periodic term for N10.

For the highest noise level N10 in which we discuss hear, the ability of learning data and the numerical result of simulation seem intuitively bad as shown in Figure 4.6. The primary reason

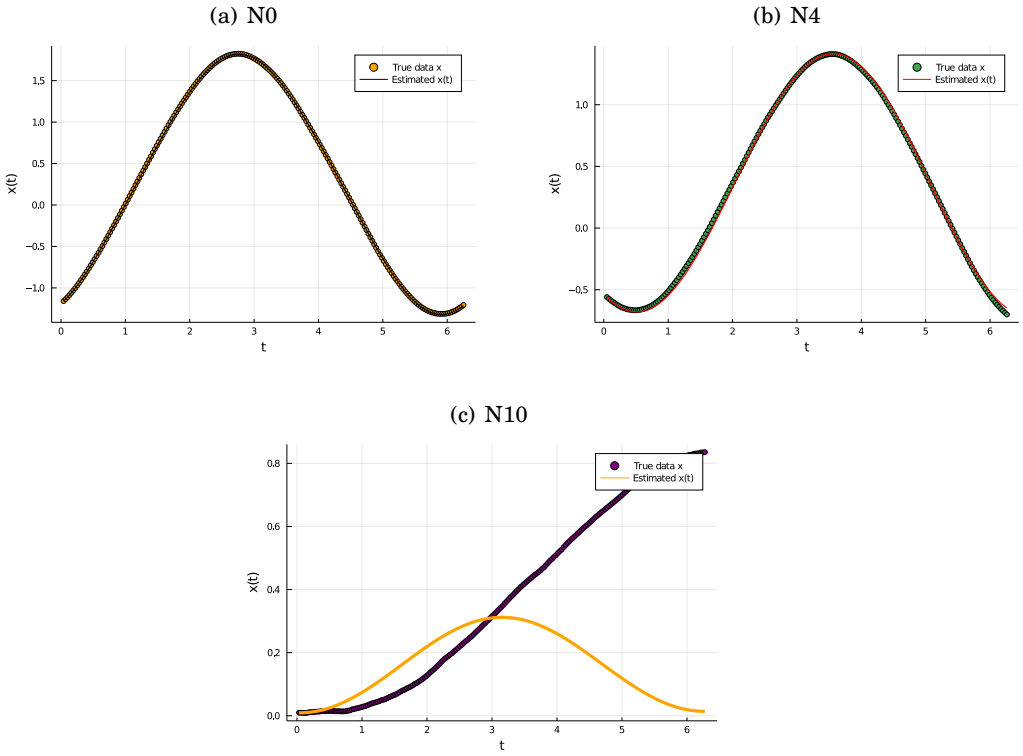


FIGURE 4.6. Comparison among approximation results at different levels of noise: (a) This figure is for noise level N0, i.e. without any additional noise, where orange circles represent the actual data and the black curve describes the corresponding estimated result from UDE approximation; (b) This figure is for noise level N4, where green circles is the actual data and the red curve shows the learning result from UDE approximation; (c) This figure is for high noise level N10, where purple circles illustrate the true data and the yellow curve provides its estimated result from UDE approximation.

of this consequence can be mainly considered as the influence from the abundant additional exterior noise. Simultaneously, the result of universal approximator has considerable distinction between N10 and N0. From another perspective, it also subjectively reflects the mathematical characteristic of the periodic property from the initial dynamic system as Equation (4.4), which can not be negligible.

Hence, we still hope UDE model can learn parts of information from the dynamical system and output further behaviours, even though the approximation result within 1-208 tests against time is disappointing. In order to find whether it is possible to predict the further trajectory in this nonlinear system or not, we extrapolate the test time series from the first 208 data samples to the first 835 data samples against time as shown in Figure 4.7. Similarly, the former data in the first 208 data samples is utilized to train neural network in UDE model for learning the dynamical system, and further extrapolation result is compared by the true data collected from

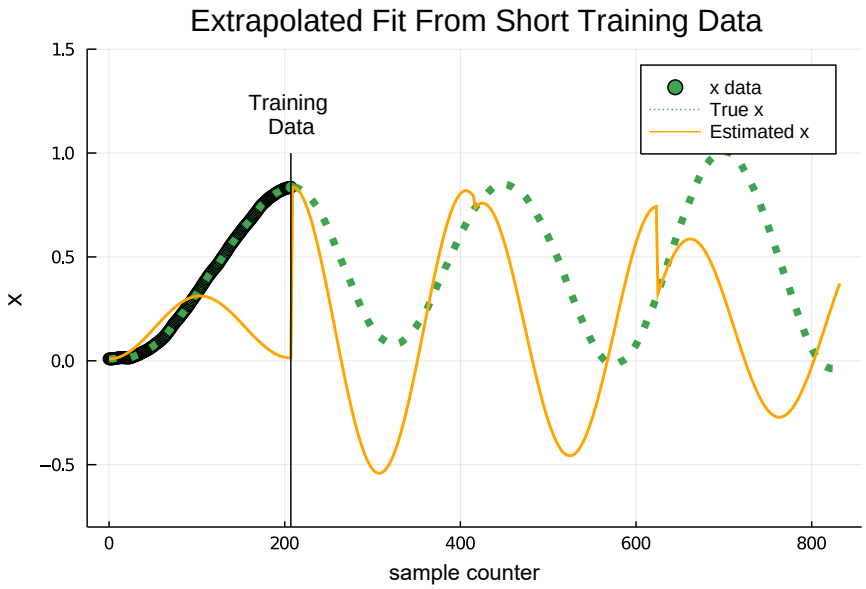


FIGURE 4.7. Extrapolation result with N10 in the first 835 data samples: The green big circles and small squares provide true data during the first 208 data samples and further the first 835 samples respectively, and the orange curve represents the extrapolation result by the first 208 data samples.

experimental device.

It can be observed from Figure 4.7 that both of the numerical prediction by UDE and actual data selected from experimental rig have similar fluctuation tendency although their orbits against time do not perfectly overlap with each other, which is a surprising outcome. Moreover, closer inspection of the result curve reveals that there exists several sharp skips near the peaks. While discussing the reasons of this phenomenon, besides the effect from high level of noise, another possible explanation for this issue is that there exists non-negligible error caused by calculating approximate derivative through the experiments described in Equation (4.6). Since the truncation error of backward difference for each calculation step is $O(h_i)$ where $h_i = |t^{(i+1)} - t^{(i)}|$ is the subinterval width, the experimental error from computing may become astonishing if the interval width h_i is not relatively eligible.

In the final part of the experiment, the errors of numerical results with respect to the three different level of noise are compared as presented in Figure 4.8. As the noise level becomes higher, the corresponding average experimental error obviously increases. Specifically, the error for N0 is the most stable which is almost controlled from 10^{-3} to 10^{-2}, while the fluctuations of error for N4 and N10 are noticeably more tempestuous and the margins of error are wider. The result provides the fact that additional noise at high level is able to produce negative effect on learning data and prediction consequence by UDE model.

To summarize the experiment in the presence of noise for the specific dynamical model, these

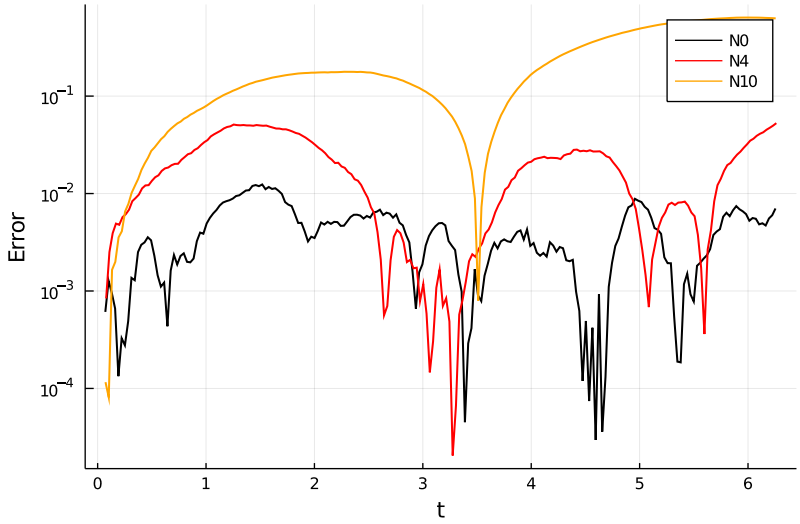


FIGURE 4.8. Experimental error in the presence of noise for N0 in black, N4 in red and N10 in yellow.

results suggest that there is an association between noise level and numerical experiments. As for the low level of additional noise, the simulation is quite superior and accurate. It can be seen that UDE model has strong capability of learning data and improve the constructed scientific model for the Duffing-like system by identifying the possible mathematical items. As for high level of noise, even though the error of prediction is unacceptable and unstable, UDE model is capable to catch parts of major dynamical characteristics and infer some further trajectories or behaviours.

CHAPTER 5

SUMMARY AND FUTURE WORK

This work is to explore to numerically simulate some specific nonlinear dynamics systems, e.g. Duffing-like systems, in order to capture their main mechanical behaviours by training machine learning embedded differential equations. Throughout the whole research, we focused on introducing universal differential equations for identifying dynamical system models, which mainly involves several crucial algorithmic techniques and fundamental theories.

In Chapter 1, we introduce the academic background of our work at first. Secondly, we state the topic discussed in this paper and present potential stakeholder relevance, specifically, amongst the fields of related industries referring to dynamics. And in the final part, the outline can be found.

In Chapter 2, we briefly demonstrate the historic research on ANN and show the basic mathematical model of ANN.

In Chapter 3, the relevant theories and techniques can be found. Firstly, we introduce SInDy algorithm proposed recently to identify the algebraic functions in differential models. Secondly, we present universal approximation theorem, which is the fundamental theory of neural network acting as an universal approximator. Then, some derivative-based methods are demonstrated to optimize the loss function while training data. Finally, we introduce our algorithmic sources from Julia SciML.

In Chapter 5, we show our experimental results on Duffing system and Duffing-like system respectively. As the initial step, a preliminary numerical experiment for a motivating example, duffing equation, has been operated to promise the efficiency and accuracy of the UDE technique in the case of missing different terms and observing the results to give appropriate adjustment for future experiments. Secondly, the similar numerical methods are utilized into specific datasets collecting from a nonlinear dynamical system. From the training results, prominent performance

can be found for almost correct approximation when there exists a low level of noise, as well as excellent extrapolation from training data in a short time series. The results from training UDE may ignore the small terms, especially while considering time variable. However, the error becomes unacceptable at the presence of high level of noise.

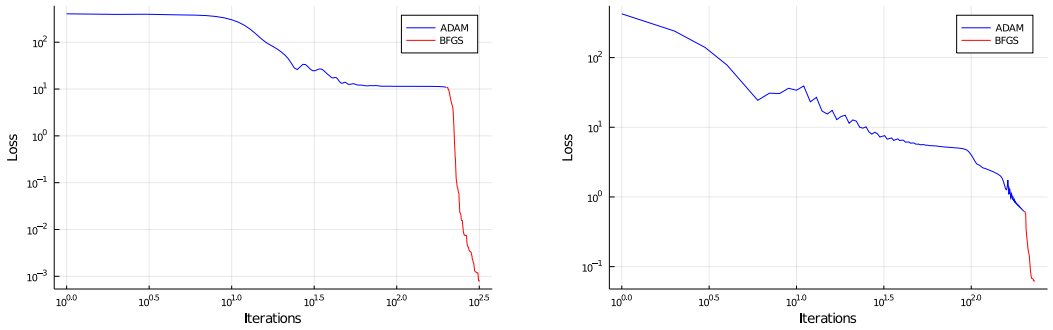
To sum up, the advantages and disadvantages of the combination of ANN and SInDy have been mentioned that ANN is able to approximate data precisely and utilize the previously known scientific knowledge but is unable to output direct interpretability of the basic dynamical models in physical worlds, while SInDy can discover the governing equations in the case of sparse dynamics but could not consider the original domain information. It is a newly developing mathematical methodology to link between machine learning models and scientific models described by differential equations, which enables computational programming to accurately train time-dependent data from dynamical systems and extrapolate future evolutionary tendencies.

For the future research about this project, several attemptable methods can be taken into the experiments, such as adding more neurons for deeper training and collecting more testing data for identification. Meanwhile, we can try other kinds of derivative-based methods during the experimental process and different mathematical models to capture the mechanical behaviours. Furthermore, we can also train different data sources and more complex dynamical systems.

APPENDIX

(a) Sigmoid function

(b) Hyperbolic Tangent function



(c) Softplus function

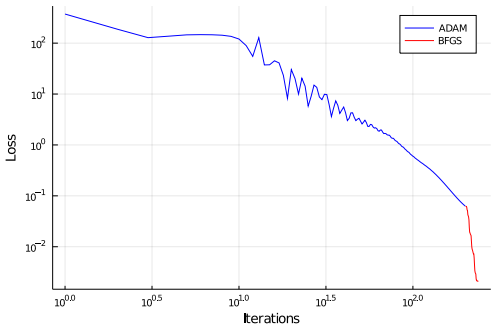


FIGURE A.1. Compare loss functions among different activation functions: (a) Sigmoid function (b) Hyperbolic Tangent function (c) Softplus function.

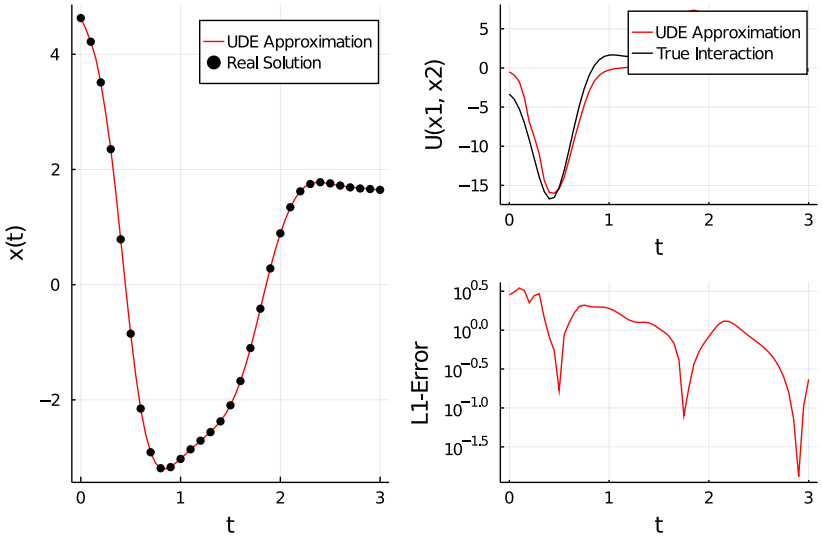


FIGURE A.2. Experimental result by Sigmoid activation function.

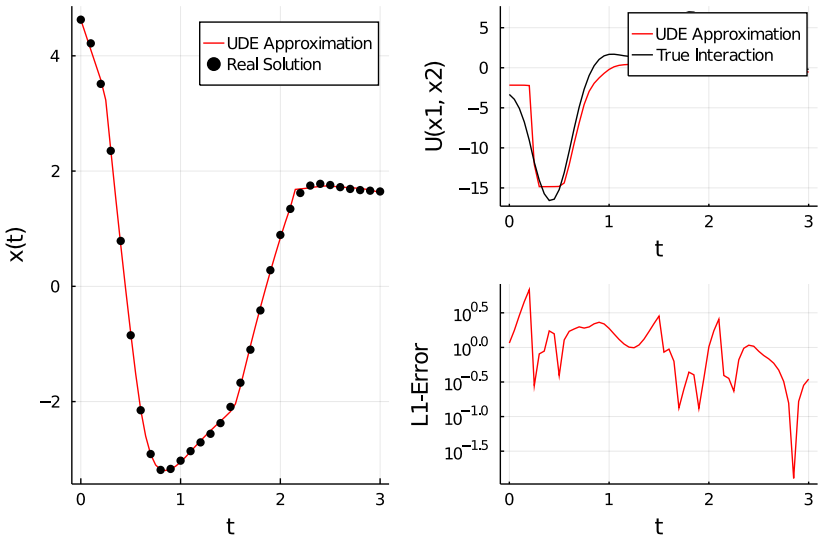


FIGURE A.3. Experimental result by Hyperbolic Tangent activation function.

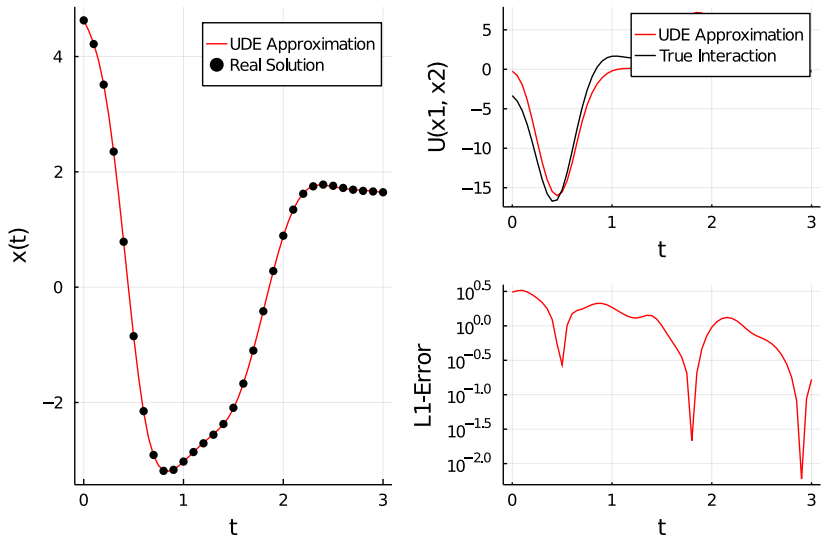


FIGURE A.4. Experimental result by Softplus activation function.

BIBLIOGRAPHY

[1] *Ann described by figure: https://en.wikipedia.org/wiki/artificial_neural_network*.

[2] *Julia platform: https://julialang.org/*.

[3] *Scientific machine learning software: https://sciml.ai/*.

[4] C. BAILER-JONES, D. MACKAY, AND P. WITHERS, *A recurrent neural network for modelling dynamical systems*, Network, 9 (1998), pp. 531–47.

[5] D. BALLARD, *Modular learning in neural networks*, (1987), pp. 279–284.

[6] D. BARTON AND J. SIEBER, *Systematic experimental exploration of bifurcations with noninvasive control*, Physical Review E, 87 (2013), p. 052916.

[7] S. BEREGI, D. BARTON, D. REZGUI, AND S. NEILD, *Robustness of nonlinear parameter identification in the presence of process noise using control-based continuation*, Dynamical Systems, 104 (2020), pp. 885–900.

[8] S. BILLINGS, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*, John Wiley and Sons, Incorporated, 2013.

[9] C. BISHOP, *Pattern Recognition and Machine Learning*, Springer (India) Private Limited, 2013.

[10] S. BRUNTON, J. PROCTOR, AND J. KUTZ, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the National Academy of Sciences, 113 (2016), pp. 3932–3937.

[11] A. BRYSON, *A gradient method for optimizing multi-stage allocation processes*, Proceedings of the Harvard University Symposium on digital computers and their applications, (1961).

[12] D. BUEDE AND W. MILLER, *The Engineering Design of Systems: Models and Methods*, Wiley, 2016.

[13] H. CAO, Y. C. L. KANG, AND J. YU, *Evolutionary modeling of systems of ordinary differential equations with genetic programming*, Genetic Programming and Evolvable Machines, 1 (2000), pp. 309–337.

[14] G. CYBENKO, *Approximation by superpositions of a sigmoidal function*, Mathematics of Control, Signals, and Systems, 2 (1989), p. 303–314.

[15] A. DAVID, *Control-based continuation: Bifurcation and stability analysis for physical experiments*, Mechanical Systems and Signal Processing, 84 (2017), pp. 54–64.

[16] W. DAVIDON, *Variable metric method for minimization*, (1959).

[17] P. DEBYE, *Näherungsformeln für die zylinderfunktionen für große werte des arguments und unbeschränkt veränderliche werte des index*, Mathematische Annalen, 67, pp. 535–558.

[18] P. DIEDERIK AND B. JIMMY, *Adam: A method for stochastic optimization*, CoRR, abs/1412.6980 (2015).

[19] P. DRAZIN, R. JOHNSON, D. CRIGHTON, M. ABLOWITZ, S. DAVIS, E. HINCH, A. ISERLES, J. OCKENDON, AND P. OLVER, *Solitons: An Introduction*, Cambridge Texts in Applied Mathematics, Cambridge University Press, 1989.

[20] J. DUCHI, E. HAZAN, AND Y. SINGER, *Adaptive subgradient methods for online learning and stochastic optimization*, Journal of Machine Learning Research, 12 (2011), pp. 2121–2159.

[21] B. FARLEY AND W. CLARK, *Simulation of self-organizing systems by digital computer*, Transactions of the IRE Professional Group on Information Theory, 4 (1954), pp. 76–84.

[22] R. FLETCHER, *Practical methods of optimization*, 1988.

[23] M. HASSOUN AND A. HASSOUN, *Fundamentals of Artificial Neural Networks*, A Bradford book, MIT Press, 1995.

[24] S. HAYKIN, S. HAYKIN, AND S. HAYKIN, *Neural Networks: A Comprehensive Foundation*, International edition, Prentice Hall, 1999.

[25] D. HEBB, *The Organization of Behavior: A Neuropsychological Theory*, Taylor & Francis, 2005.

[26] M. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, Journal of research of the National Bureau of Standards, 49 (1952), pp. 409–435.

[27] S. HOCHREITER AND J. SCHMIDHUBER, *Long short-term memory*, Neural Computation, 9 (1997), pp. 1735–1780.

[28] K. HORNIK, M. STINCHCOMBE, AND H. WHITE, *Multilayer feedforward networks are universal approximators*, Neural Networks, 2 (1989), pp. 359–366.

[29] A. IVAKHNENKO AND V. LAPA, *Cybernetic predicting devices*, CCM Information Corporation, (1965).

[30] H. KELLEY, *Gradient theory of optimal flight paths*, ARS Journal, 30 (1960), p. 947–954.

[31] D. B. K.H. LEE AND L. RENSON, *Reduced-order modelling of flutter oscillations using normal forms and scientific machine learning*, arXiv: Dynamical Systems, 1 (2020).

[32] M. LESHNO, V. LIN, A. PINKUS, AND S. SCHOCKEN, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Networks, 6 (1993), pp. 861–867.

[33] H. LIN AND S. JEGELKA, *Resnet with one-neuron hidden layers is a universal approximator*, in NeurIPS, 2018.

[34] S. LINNAINMAA, *The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors*, Proceedings of University of Helsinki, (1970).

[35] Z. LU, H. PU, F. WANG, Z. HU, AND L. WANG, *The expressive power of neural networks: A view from the width*, in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., vol. 30, Curran Associates, Inc., 2017.

[36] P. P. M. RAISSI AND G. KARNIADAKIS, *Inferring solutions of differential equations using noisy multi-fidelity data*, Journal of Computational Physics, 335 (2017), pp. 736–746.

[37] N. MANGAN, S. BRUNTON, J. PROCTOR, AND J. KUTZ, *Inferring biological networks by sparse identification of nonlinear dynamics*, IEEE Transactions on Molecular, Biological and Multi-Scale Communications, 2 (2016), p. 52–63.

[38] N. MANGAN, J. KUTZ, S. BRUNTON, AND J. PROCTOR, *Model selection for dynamical systems via sparse regression and information criteria*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 473 (2017), p. 20170009.

[39] W. MCCULLOCH AND W. PITTS, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics, 5 (1943), pp. 115–133.

[40] A. MEDIO AND M. LINES, *Nonlinear Dynamics: A Primer*, Cambridge University Press, 2001.

[41] T. MITCHELL, *Machine Learning*, McGraw-Hill Education, 1997.

[42] P. MOGENSEN AND A. RISETH, *Optim: A mathematical optimization package for julia*, The Journal of Open Source Software, (2018).

[43] K. NARENDRA AND K. PARTHASARATHY, *Neural networks and dynamical systems*, International Journal of Approximate Reasoning, 6 (1992), pp. 109–131.

[44] A. NAYFEH AND B. BALACHANDRAN, *Applied Nonlinear Dynamics: Analytical, Computational and Experimental Methods*, Wiley, 2008.

[45] R. NEAL, *Bayesian Learning for Neural Networks*, Springer New York, 1996.

[46] H. NIELSEN, *Theory of the backpropagation neural network*, vol. 1, 1989.

[47] H. OWHADI, *Bayesian numerical homogenization*, SIAM Multiscale Modeling and Simulation, 13 (2015), pp. 812–828.

[48] H. OWHADI, C. SCOVEL, AND T. SULLIVAN, *Brittleness of bayesian inference under finite information in a continuous world*, Electronic Journal of Statistics, 9 (2015), pp. 1–79.

[49] A. PINKUS, *Approximation theory of the mlp model in neural networks*, Acta Numerica, 8 (1999), p. 143–195.

[50] M. QUADE, M. ABEL, K. SHAFI, R. NIVEN, AND B. NOACK, *Prediction of dynamical systems by symbolic regression*, Physical review. E, 94(1-1) (2016), p. 012214.

[51] C. RACKAUCKAS, Y. MA, J. MARTENSEN, C. WARNER, K. ZUBOV, R. SUPEKAR, D. SKINNER, AND A. RAMADHAN, *Universal differential equations for scientific machine learning*, ArXiv, 1 (2020).

[52] M. RAISSI AND G. KARNIADAKIS, *Hidden physics models: Machine learning of nonlinear partial differential equations*, Journal of Computational Physics, 357 (2017).

[53] M. RAISSI, P. PERDIKARIS, AND G. KARNIADAKIS, *Numerical gaussian processes for time-dependent and nonlinear partial differential equations*, SIAM Journal on Scientific Computing, 40 (2018), pp. A172–A198.

[54] M. RAISSIA, P. PERDIKARISB, AND G. KARNIADAKISA, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2018).

[55] A. RAMM AND N. HOANG, *Dynamical Systems Method and Applications: Theoretical Developments and Numerical Examples*, Wiley, 2013.

[56] C. RASMUSSEN AND C. WILLIAMS, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[57] D. RUMELHART, G. HINTON, AND R. WILLIAMS, *Learning Internal Representations by Error Propagation*, MIT Press, Cambridge, MA, USA, 1986.

[58] B. SPEELPENNING, *Compiling fast partial derivatives of functions given by algorithms*, PhD thesis, Department of Computer Science, University of Illinois Urbana-Champaign, United States, 1980.

[59] S. STROGATZ AND M. DICHTER, *Nonlinear Dynamics and Chaos, 2nd ed. SET with Student Solutions Manual*, Avalon Publishing, 2016.

[60] T. TIELEMAN AND G. HINTON, *Lecture 6.5 - rmsprop, coursera: Neural networks for machine learning*, Technical report, (2012).

[61] Y. TIUMENTSEV AND M. EGORCHEV, *Neural Network Modeling and Identification of Dynamical Systems*, Elsevier Science, 2019.

[62] M. TOPLISS, C. TOOMER, AND D. HILLS, *Rapid design space approximation for two-dimensional transonic aerofoil design*, Journal of Aircraft, 33 (1996).

[63] X. WU, *Preliminary report: Learning nonlinear dynamics from experimental data*, (2021).

[64] W. YI-JEN AND L. CHIN-TENG, *Runge-kutta neural network for identification of dynamical systems in high accuracy*, IEEE Transactions on Neural Networks, 9 (1998), pp. 294–307.