Deep Learning for Modelling and Simulations in Physics

陈, 钰涵

(Degree)
博士（システム情報学）

(Date of Degree)
2023-09-25

(Date of Publication)
2024-09-01

(Resource Type)
doctoral thesis

(Report Number)
甲第8745号

(URL)
https://hdl.handle.net/20.500.14094/0100485929

Doctoral Dissertation

Deep Learning for Modelling and Simulations in Physics

(物理モデリングとシミュレーションのための深層学習)

Department of Computational Science

Graduate School of System Informatics

Kobe University

CHEN YUHAN

陈 钰涵

Supervisor: Mitsuo Yokokawa

Primary Examiner: Mitsuo Yokokawa

Examiner: Akira Kageyama

Examiner: Hideki Sano

Examiner: Takaharu Yaguchi

July 11th, 2023

Abstract

CHEN YUHAN

In recent years, substantial research on the application of deep learning to the field of physical simulation has been actively explored. Physical simulation contributes to the discovery of new drugs and materials, the design of aircraft body shapes and the control of robot arms, among many other areas. Since one of the characteristics of artificial neural networks is their strong applicability, they are capable of learning physical systems and simulating physical phenomena. This thesis focuses on deep learning for physical modeling and simulation. Specifically, we develop new deep physical models and their extensions along with theoretical analysis.

In the first study, we applied artificial neural networks to a secret communication system for color images. Hereby, the neural network is applied to learn a nonlinear boundary condition of a chaotic wave equation, making it difficult to steal information. In the previous research, a secret communication system using wave equations with a nonlinear boundary condition causing chaotic oscillations has been considered. The wave equation with the van der Pol boundary condition is used to form chaotic oscillations by satisfying the parameter conditions on the 1-dimensional bounded interval. The application of the chaotic distribution system to a secret communication system extends the transmitted signal from a scalar to vector values, which makes the system easy to be applied to the encryption and decryption of image signals. However, the system has the disadvantage that the number of parameters of the van der Pol boundary condition is so few that the communication image can be easily stolen.Therefore, in order to improve the security and concealment of the confidential communication system, this study models the boundary conditions using deep neural networks, which can be regarded as a black box.

Other studies are about deep physical models. In terms of deep physical modeling, Lagrangian neural networks and Hamiltonian neural networks were developed based on Lagrangian mechanics and Hamiltonian mechanics, which can guarantee the physical properties of physical systems. Firstly, the Hamiltonian neural network is a powerful method; however, theoretical studies are limited. In this study, by combining the statistical learning theory and KAM theory, we provide a theoretical analysis of the behavior of Hamiltonian neural networks when the learning error is not com-

pletely zero. A Hamiltonian neural network with non-zero errors can be considered as a perturbation from the true dynamics, and the perturbation theory of the Hamilton equation is widely known as the KAM theory. To apply the KAM theory, we provide a generalization error bound for Hamiltonian neural networks by deriving an estimate of the covering number of the gradient of the multi-layer perceptron, which is the key ingredient of the model. This error bound gives an L^∞ bound on the Hamiltonian that is required in the application of the KAM theory.

Secondly, although existing neural network models are very promising, the commonly used representation of the Hamilton equation uses the generalized momenta, which are generally unknown, and this causes difficulty in applying the models to real data. Meanwhile, Hamiltonian equations also have a coordinate-free expression that is expressed by using the symplectic 2-form. In this study, we propose the neural symplectic form that learns the symplectic form from data using neural networks, thereby providing a method for learning Hamiltonian equations from data represented in general coordinate systems, which are not limited to the generalized coordinates and the generalized momenta.

Thirdly, such deep physical models should be used in physical simulations, which means that discretization of the models by numerical integrators are necessary. We propose structure-preserving numerical integrators for Hamiltonian neural networks as well as for the neural symplectic form, respectively. It is known that when general numerical integrators are used for discretization, the energy conservation law and other laws of physics are destroyed. Structure-preserving numerical methods such as the variational integrator are effective to address this problem.

Last but not least, we propose a method for super resolution. Super resolution has been widely used to enhance the resolution of images in recent years, thereby obtaining clearer images. In the field of physics simulation, super-resolution technique has also been actively developed. However, existing methods based on standard neural networks work on a single scale factor, making it unsuitable for cases where arbitrary super-resolution scale factors are required. Meanwhile, DeepONet is a deep learning framework capable of learning operators between function spaces. In this study, we propose a DeepONet-based approach to super resolution of numerical solutions of nonlinear partial differential equations. Some experimental results are also provided.

Acknowledgement

The year 2023 marks my fifth year as a student at Kobe University, and I will be completing my PhD in this year. From a research student to PhD, I have gained a lot of unexpected growth and unanticipated research results.

Reviewing my research life, the person I would like to thank the most is Professor Takaharu Yaguchi. Since I was assigned to this laboratory, Professor Takaharu Yaguchi has given me countless guidance and support. He taught me not only technical knowledge but also a lot of experience with regard to research. Every time I created a presentation, Professor Takaharu Yaguchi was very attentive and strict in teaching me, even the details, which made me grow a lot. He was also very patient in the discussions and talks related to the thesis, and I am very grateful for all this. In addition to academic life, whenever I encountered difficulties in my life, he listened and comforted me, even if it was something trivial, which gave me a lot of emotional support and courage. When I recall that after my grandmother passed away, I was unable to return home because of the epidemic of coronavirus, it was also my professor who looked after my feelings and accompanied me during this difficult time. I am indescribably grateful to my professor.

Professor Mitsuo Yokokawa of the Basic of Computational Science Lab, who is also the supervisor of this thesis, advised me on my research and the presentations made in the laboratory seminar. At the same time, his optimistic personality had a positive impact on me. I would like to thank Professor Hideki Sano of the Graduate School of System Informatics, Kobe University, for accepting to review this thesis following my master's thesis, and for his various useful comments and suggestions during the degree review process. I would like to thank Professor Akira Kageyama of the Graduate School of System Informatics, Kobe University, for accepting to review this thesis. I would also like to thank Professor Takashi Matsubara of the Graduate School of Engineering Science, Osaka University, for his many useful suggestions and useful advice in various research.

Also, I would like to thank my laboratory mates Mr. Kohei Ohkawa, Mr. Yujiro Takenaka, et al. who gave me a lot of enthusiastic help when I was in trouble, which makes me very happy memories of my research life. In particular, I am grateful to Ms. Mizuka Komatsu, who was my tutor and did not make me feel lonely. I would

also like to thank Ms. Baige Xu for coming along and I am very happy that she has been an unexpected surprise in my study abroad life.

Last but not least, I would like to thank my parents, Mrs. Lixia Hu and Mr. Chunsheng Chen, who acted as my safe haven, for supporting me in studying in Japan. Without their encouragement and support behind me, I would not have been able to complete my studies alone in a foreign country.

Once again, I would like to thank everyone who has been present in my life during these five years of study abroad. I would like to thank them for helping me directly or indirectly to complete this thesis.

Contents

Chapter 1 Introduction

Physical simulation has been performed in a wide range of science and engineering applications. However, for phenomena in which the model is unknown, the model must be constructed from data. Building deep learning models of physical phenomena enables the simulation of physical phenomena for which the model is unknown. Hence, data-driven modelling methods, in particular models based on deep learning, have attracted much attention in recent years.

The main research of this thesis is a combination of deep learning and physics. The aim is to explore the applicability of artificial neural networks in physics and to develop deep physical models with more general applicability. In particular, we aim to develop methods that preserve physical properties. We also provide theoretical analysis of these models as well as numerical experiments.

The main work of this thesis is as follows:
(1) As an application of deep physical modelling, we apply deep neural networks to image encryption. Deep neural networks strengthen the security performance of chaotic synchronous systems by learning the van der Pol boundary conditions. We have confirmed that deep neural networks have good approximation ability while also effectively acting as black boxes via numerical experiments.
(2) In the field of data-driven physical model discovery, we have combined deep learning with geometric mechanics theory to construct models that retain physical properties and theoretically analyze the properties of existing models and the proposed model. Further to this, we have also developed numerical integrators for the discretization of deep physical models. In this thesis, we refer to neural network models that learn equations of motion that perform physical phenomena as deep physical models.

1.1 Deep Learning in Physics

In the last few years, deep learning has led to very good performance on a variety of problems, such as visual recognition [58,63], speech recognition [23,82] and natural language processing [5, 59, 113]. Although deep learning is promising, the training of deep neural networks often requires large amounts of data, and in many scientific problems, data is difficult to obtain. Physics is a well-established field that has been studied theoretically and experimentally for a very long time. This theory can explain the essence of the behavior of both natural phenomena and artificial objects. We can design better machine learning models by embedding physical knowledge, such as physical laws, PDEs, or simplified mathematical models, into neural networks. In addition to that these models can automatically satisfy some conservation laws, the models can be trained faster and achieve better accuracy. Therefore, the development of new generation models that integrate with physical laws has become a promising area of machine learning research [44, 45].

Deep neural networks have been in fact applied into physics [85]. [107] present a novel deep-learning-based robust nonlinear controller (Neural Lander), that combines a nominal dynamics model with a deep neural network that learns high-order interactions. [72,135] developed models based on Lagrangian and Hamiltonian mechanics in the modeling of underlying system dynamics respectively. In these papers, it was concluded that due to the integration of physics knowledge, physically informed machine learning models enjoy better physical plausibility, higher data efficiency and greater generalisation than simple neural network models. In addition, a hybrid in situ – in silico algorithm, called physics-aware training, was introduced which applies backpropagation to training controlled physical systems [123].

One approach to implementing deep learning is to use Convolutional Neural Networks (CNN) [62,63]. Convolutional neural networks are modeled to simulate human visual perception. Convolutional neural networks are trained by feeding the network with pre-processed images, and by extracting important features hidden in these inputs, the algorithm recognizes desirable patterns and relevant points. Convolutional neural networks have been applied to physics. For example, they are used for particle recognition and classification as well as for particle trajectory reconstruction [83,128]. Convolutional neural networks have been used for outer-Earth exploration. In astro-

physics, some scientists are developing Convolutional neural networks that can discover new gravitational lensing [11]. Quantum convolutional neural network (QCNN) is used into the classification of high-energy physics events. This model has quantum architecture that demonstrates an advantage of learning faster than the classical Convolutional neural networks under a similar number of parameters.

The Physics Informed Neural Network (PINN), which embeds with physical information to a deep fully-connected neural networks [92, 93]. This model is an application of the traditional scientific computing, particularly for solving problems related to partial differential equations, including equation solving, parameter inversion, model discovery, control and optimisation [22, 91, 117, 122]. Physical principles in Physics Informed Neural Networks provide theoretical insight and illuminate the underlying mechanisms, in addition to enhancing the trainability and generalisation performance of deep learning models [117]. Meanwhile, for modeling and forecasting multi-physical systems, a novel framework named physics-informed convolutional network (PICN) is suggested from a CNN perspective [108]. As a variant of Physics Informed Neural Networks, another approach is physics-constrained neural networks (PCNNs) [136], in which it is reported that the amount of training data can be significantly required. However, the weights of different losses from data and physical constraints are adjusted empirically in physics-constrained neural networks. In [69], a new physics-constrained neural network with the minimax architecture (PCNN-MM) is proposed so that the weights of different losses can be adjusted systematically. In [77], a neural network model that extend the physics-informed neural networks to be trained with multi-fidelity data sets (MPINNs). In addition, other extensions that include hp-VPINN [54] and CPINN [48] have been also proposed.

In addition to the above proposals of new models, the Physics Informed Neural Network has an extremely wide range of physical and engineering applications. [10, 41] studied the application of Physics Informed Neural Networks to various heat transfer problems and demonstrated that the model has broad applications in materials science. [12] proposed a method to directly predict the expected flow fields based on given flow conditions and geometry shapes, which saves computational time. In [109], the Physics Informed Neural Network is trained to solve the problem of identifying and characterizing a surface-breaking crack in a metal plate.

1.2 Deep Physical Models

1.2.1 Analytical mechanics

Among applications of neural networks for physics, the model discovery, in which the fundamental equations of classical mechanics are modeled by neural networks, have been actively studied. Analytical mechanics, which is a theory of classical mechanics, has two dominant branches, Lagrangian mechanics and Hamiltonian mechanics [1, 4, 73]. The equations of motion of Lagrangian mechanics are called the Euler–Lagrange equation, for which several neural network models were proposed [20, 72]. The Euler–Lagrange equation is defined as

$$\frac{\partial \mathcal{L}}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}} = 0, \tag{1.1}$$

where q is the position and $\dot{q}$ is the time derivative of q. This equation is known to equivalent to Newton's equation of motion.

1.2.2 Deep learning models for physical model discovery

Lagrangian neural networks use neural networks to model the Lagrangian L and derive the equations of motion from it as the following form:

$$\frac{\partial \mathcal{L}_{\mathrm{NN}}}{\partial q} = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}_{\mathrm{NN}}}{\partial \dot{q}}. \tag{1.2}$$

However, because the theory defines systems on tangent bundles, it is restricted to systems with a specific symplectic structure and is only equivalent to a part of Hamiltonian systems. In Lagrangian mechanics, the equations are described using the state variables and the time derivatives of them. This feature of Lagrangian mechanics makes it easy to prepare the data necessary for learning.

On the other hand, Hamiltonian mechanics can describe more general equations which are not covered by Lagrangian mechanics. In geometry, the Hamiltonian equation is defined on the cotangent bundle, which is an example of a symplectic manifold. The governing equation of Hamiltonian mechanics is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} O & I \\ -I & O \end{pmatrix}\begin{pmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{pmatrix}. \tag{1.3}$$

In this equation, q and p represent generalized coordinates and generalized momenta, respectively. H is a function that depends on q and p and represents the total energy of the system. This function H is also called the Hamiltonian of the system. In recent years, there has been a lot of research on predicting the corresponding physical phenomena by learning the energy function H in such equations with a neural network H_{NN} (e.g., [19, 20, 38, 75, 135]). Among them, the Hamiltonian neural network is a powerful method, which is defined as the following form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} O & I \\ -I & O \end{pmatrix} \nabla H_{\mathrm{NN}}. \tag{1.4}$$

As the Hamilton equation has the energy conservation law, the Hamiltonian neural network also guarantees this conservation law. This conservation law is known to greatly improve the performance in long-term forecasting. In addition, in the last four years, research on deep physical models still has intensively performed, and numerous extensions have been proposed, including the DGNet [75] and VIN [99], as well as other models discretized in the time direction.

In terms of theoretical studies of deep physical models, the universal approximation theorem for discrete-time neural network models are provided in SympNet [51] for the Hamilton equation. However, theoretical analysis for deep physical models has not progressed sufficiently.

1.2.3 Discretization of deep physical models for simulations

These deep physical models are used in physical simulations, which means that discretization of the models is required. Therefore, models discretized in the time direction are particularly useful, as such models do not require a further discretization for simulations. On the other hand, discrete-time models have a disadvantage of being unable to be simulated with time steps other than that used in the training process. As a result, developing a way to discretize continuous-time deep physical models for simulation is important. In addition, discretization methods of deep physical models are also useful for designing discrete-time models.

The numerical integrators that preserve the mechanical structure are called the structure-preserving integrators or the geometric integrators. Among them, symplectic integrators are the most important. They are designed so that the symplectic

form is preserved, thereby preserving the physical properties. One way to design a symplectic integrator is the variational integrator for Lagrangian mechanics [74]. By discretizing the variational principle, the variational integrator discretizes the Euler–Lagrange equation while preserving various conservation laws.

In addition to the discrete-time models mentioned above, there have been many studies of discrete-time models, such as those using Gaussian process regression [95]. As for the variational integrators, VIN [99] and VIGN [24] are known as discrete-time models in Lagrangian formalism. However, there has been no research on variational integrators for neural network models that represent the Hamilton equation.

1.3 Aim of This Thesis

The goal of this thesis is to develop novel deep physics models along with theoretical analysis, thereby extending the applicability of artificial neural networks in physics. In this thesis, we focus on the following two directions.

Firstly, we focus on the application of deep physical modelling to improve a chaotic-synchronization-based encryption system of images. Specifically, the purpose of this study is to establish a more stable and secure communication system by improving the method reported in [101]. The reason for this is that the number of parameters in the confidential communication system is too small, and hence the image information can be easily stolen; in fact, even if only one parameter is attacked, there is a certain chance that the original image can be seen in its entirety. In order to improve the security of the system based on the original concept, the development of a new method that includes a chaotic system with a larger number of independent parameters is necessary. We addressed this problem by black-boxing the boundary conditions of the wave equations by approximating them with deep neural networks. We also observed whether they are certainly able to learn the chaotic behaviours. In fact, neural networks have a complex structure and are therefore difficult to steal all the information, i.e., parameters, thus increasing the security of the encryption system. Our results of this research have been published in [18].

Secondly, the Hamiltonian neural network is a deep physical model represented using the canonical coordinate system, which means that the preparation of data is difficult. As a solution to this problem, we propose the neural symplectic form, which

is not only able to model equations on a general coordinate system but also to extract unknown Hamiltonian structures hidden in the data. This research was published in [16]. Next, we also propose numerical integrators for Hamiltonian neural networks and the neural symplectic form, respectively. The aim is to discretize the model without losing physical properties when using these models for physical simulations. This research was published in [132].

For theoretical studies in deep physics models, to the best of our knowledge, theoretical analysis of such models has not been performed sufficiently. Therefore, we provide a theoretical analysis of the behaviour of Hamiltonian neural networks when the learning error is not completely zero. The stability of celestial systems and the recursive nature of physical phenomena greatly affect the usefulness of deep physics models. Thus, we performed a theoretical analysis of this using KAM Theory and statistical learning. This research is published in [17].

In addition, super resolution is a technique for increasing the resolution of images and films. When performing super resolution for physical simulations, there are certain drawbacks in the existing research; the standard neural network-based approach works on a single scale factor, which makes it unsuitable for situations where an arbitrary super-resolution scale factor is required. We propose to use DeepONet's network structure for the super resolution of numerical solutions of partial differential equations. Preliminary results of this research is currently under review.

1.4 Organization of This Thesis

In this section, we describe the organization of this thesis. In Chapter 2, we use neural networks to learn the boundary conditions of the chaotic wave equations, thereby improving the security of image transmission encryption systems. First, we outline the background related to the encryption systems, then we present the details of our proposed method, followed by numerical experiments on grayscale images as well as colour images, respectively. The Lyapunov exponent, which is a numerical value used to judge whether a system is chaotic, was also computed.

In Chapter 3, we present our proposed deep physics model, the neural symplectic form. Firstly, we summarise and compare existing deep physics models such as Hamiltonian neural networks, Lagrangian neural networks and skew-symmetric ma-

trix. Secondly, we explain the theory of this study, and describe the structure of our proposed model in detail. Finally, we conduct comparative numerical experiments, using a mass-spring system, the Lotka–Volterra equation, and double pendulums as experimental subjects.

In Chapter 4, we present a theoretical analysis of Hamiltonian neural networks with non-zero training loss. We first provide an overview of the relevant research. Then, we give a brief introduction to the Hamiltonian system and KAM theory. After that, we show the main theorems and give the corresponding proofs.

In Chapter 5, we give structure-preserving numerical integrators for Hamiltonian neural networks and the neural symplectic form, respectively. First, we give an overview of the variational integrator. Then, we illustrate the variational principle of the Hamiltonian neural network and derive a variation integrator for it. For the neural symplectic form, we also confirm the existence of the variational principle and show the derivation of the variational integrator for the neural symplectic form. In addition, we demonstrate the effectiveness of the proposed integrator using numerical experiments.

In Chapter 6, we present a method for super resolution of physics simulations for nonlinear elliptic equations. Firstly, we describe the target nonlinear elliptic equations, then we describe our proposed super resolution with DeepONet in detail, and finally, we give a numerical example.

Finally, in Chapter 7, we summarize our studies in this thesis.

Chapter 2 Secret Communication Systems Using Chaotic Wave Equations with Neural Network Boundary Conditions

This chapter aims to improve an existing confidential communication system by applying the deep physical model. More precisely, we improve a chaotic confidential communication system, in which a synchronous pair of chaotic distribution systems is used to encrypt the transmitted object, by black-boxing the the chaotic systems using a neural network to make it difficult to steal information so that the confidential communication system has a stronger secrecy effect. The following is a brief description of the proposed approach.

Firstly, following [101] the proposed approach employs the chaotic wave equation, which is an evolutional partial differential equation and hence has two axes (space and time). This feature of the equation enables the method to transmit images. Secondly, the proposed secret communication system consists of two parts, the sender and the receiver, as shown in Fig. 2.1. First, the wave equation on the sender side is under the van der Pol boundary conditions that can cause chaotic phenomena, and a deep neural network is trained so that it has the same chaotic effect. Then, the color images are encrypted by applying a certain nonlinear transformation together with the solutions to the wave equations. The nonlinear transformation is designed so that the inverse of it is computable if the solutions to the wave equations, which are essentially the secret keys, are known. Second, a chaotic synchronization system is established at the receiver side to obtain the decrypted recovered images by inverting the function.

The innovation of this study is to use a deep neural network that approximates the boundary conditions yielding the chaotic oscillations, which addresses the problem

that the previous system is easier to be cracked and makes it easier for the information to be stolen due to too few parameters of the van der Pol boundary condition when the original chaotic synchronization system is used alone. In fact, in the wave equation with the van der Pol boundary conditions, the only three parameters a, β and η are required (see Section 2.2 for the specific expressions). Suppose that a hacker now steals the specific value of β, and the other two (a and η) still exist in a hidden state. Suppose also that this person makes an effort to guess the values of a and η and tries to break the encrypted information. In Fig. 2.2, we can see that the possibility exists that the whole information of the portrait can be seen to some extent.

Although it is known that deep learning techniques are also under threat from security attacks, a stealer will face greater implementation difficulties than a brute force attack when parameters are partially compromised. More notably, the way of the proposed approach which corresponds to the neural network structure with the boundary conditions at the left end and the right end will also increase the difficulty for the stealer to crack.

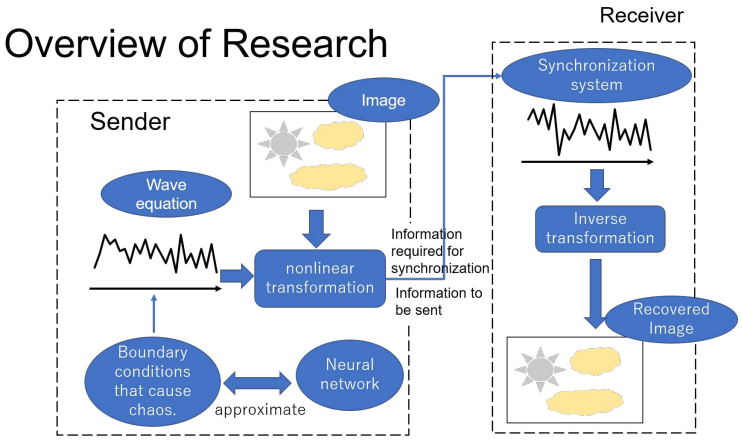


Fig.2.1 A diagrammatic representation of the research ideas and methods used in this study. It succinctly shows the overall structure of the confidential communication system as well as the design principles, in which the input portrait was created manually by the author and is only used here as an example to assist in illustration.

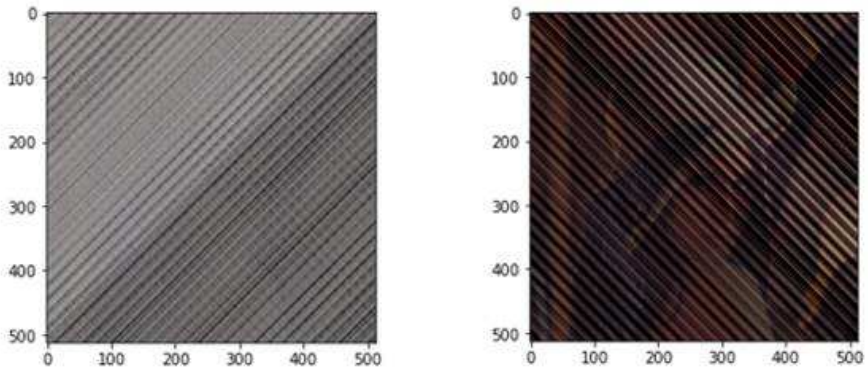


Fig.2.2 The modulated secrecy image and the restored image after the leakage of some of the parameters of the confidential communication system. The figure on the left is the confidential information after it has been encrypted, and we can see that we have no way of knowing the information about the original image, so we can say that it was a successful secret communication. The figure on the right is the restored image after the leakage of some of the parameters of the confidential communication system; we can almost see that it is a picture of a woman wearing a hat, that is to say, the hacker can restore the transmitted information to some extent after breaking some of the parameters.

This chapter is organized as follows. In Section 2.2, the proposed method for grayscale images is described in detail, along with some numerical examples. In Section 2.3, the method is applied to color images. In Section 2.4, the encryption effects of the proposed method and the AES are tested, respectively.

2.1 Background

Innovations in information processing technology have led to the rapid development of technologies for accessing and transmitting information in recent years, which has greatly improved the convenience of our daily lives. On the other hand, a wide variety of information is exchanged in public through the internet and other media, so encryption technology has become increasingly important to protect this information.

In this study, we consider confidential communication methods using chaos synchronization as one of the encryption techniques. Applying the diversity of chaotic series, pseudo-random series can be generated with statistical properties close to those of ideal random series, which can hide the original information. The problem of designing confidential communication systems using chaotic synchronization has been studied since the 1990s. The core idea of the method is to embed communication

information into a signal that behaves chaotically. The method accomplishes the steganography and the recovery of communication information [21, 53, 56, 68, 102]. For example, by applying the Hénon mapping, a chaotic synchronous control method for discrete-time nonlinear systems can be designed with significant results [102]. A Hénon map (Hénon–Pomeau attractor/map) is a discrete-time dynamical system that can generate chaotic phenomena, with an iterative expression of:

$$\begin{cases} x_{n+1} = 1 - ax_n^2 + y_n \\ y_{n+1} = bx_n, \end{cases} \tag{2.1}$$

where a, b are parameter values. When they are set to appropriate values, the system will exhibit chaotic behaviors. In [64], the color image encryption algorithms applying scrambling-diffusion are improved by introducing the transforming-scrambling-diffusing model so that the methods have better security and cryptography characteristics. In [29], a SystemC implementation of a chaos-based crypto-processor for the AES algorithm is presented, where the properties of chaotic systems are employed to cope with the parameters of the AES algorithm. An image encryption algorithm is proposed in [3], where the Arnold chaos sequence and the modified AES algorithm are combined. In [112], a method to encrypt multiple images is proposed by a combination of a fast chaotic encryption algorithm and the AES algorithm. A family of complex variable chaotic systems are used to develop an image encryption algorithm in [70]. In [119], hyper-chaos systems and DNA sequences are combined for encrypting images, where the pseudo-random sequence generated by a hyper-chaos system is transformed into a DNA sequence to diffuse the image blocks.

However, in most of the above methods, the information on the chaotic systems must be shared for encryption and decryption to generate the chaotic sequences. In other words, the parameters of the system and the initial conditions for defining the state of the system essentially played the role of the keys. Although the algorithms are quite secure under the assumption that the information on the chaotic system can be shared safely, there remains a risk of information leakage if this is not the case. Several papers have proposed methods to address this problem by using chaotic synchronization [60, 68, 79, 84, 86, 130, 131]. The chaotic systems have positive Lyapunov exponents, which means that small differences in the initial conditions will grow exponentially. Therefore, the phenomenon of synchronization of such systems

is surprising; however, it is known that such synchronization can actually occur [87] and much attention has been focused on this phenomenon. If the chaotic systems of the receiver and the sender can be synchronized by introducing appropriate control signals, the sharing of the parameters and the initial conditions becomes unnecessary. In particular, in [68], chaos-based synchronized dynamic keys are designed and an improved chaos-based advanced encryption standard (AES) algorithm is developed so that a powerful method is proposed that solves the problem of using a static key for AES, while retaining the advantages of the chaos synchronization-based method.

In this study, we focus on the system proposed in [101], in which a certain initial boundary value problem of the linear wave equation is employed on a one-dimensional bounded interval, with a linear homogeneous boundary condition at the left end and the nonlinear boundary condition, which has a cubic nonlinearity of the van der Pol type [13, 37]. The interaction of these linear and nonlinear boundary conditions leads to chaos in the Riemann invariants (u, v) of the wave equation when the parameters satisfy certain requirements. By constructing an observer by applying the method of characteristics, the appropriate range of the feedback gain is obtained so that the convergence of the dynamics that describes the synchronization error between two mappings is ensured. Through the numerical computation, it is also confirmed that the one-dimensional wave equation with the van der Pol type boundary conditions exhibits a spatio-temporal chaotic behavior in its dynamics [65]. In [101], this chaotic vibration of the wave equation under the van der Pol boundary condition is specifically used to construct the synchronous system. There are two features of this approach—firstly, the synchronization system is easy to construct and secondly, it transmits vector-valued signals in a secure communication system [101].

On the other hand, many information security techniques are based on artificial intelligence. As the problem of image recognition is a specialty of neural networks, neural networks are often used in information security technologies to solve problems of image recognition and image analysis. Biometric technologies, such as face recognition systems and fingerprint authentication, are examples of this [81].

Meanwhile, in terms of security performance, deep learning also does not have the complete capability to guarantee absolute security and privacy. There are many types of attacks that are made on deep learning, such as Causative Attacks, Exploratory Attacks and Indiscriminate Attacks [115]. With these attacks, the model information

or the knowledge of the training data can be extracted; these are known as model inversion, model extraction and membership inference, where the attackers steal the training data and produce the expected results, or provide incorrect training data. This means that the attackers may have the ability to change the inputs to the training data, which becomes the reason for parameter changes in the learning model, leading to a significant decrease in the performance of the subsequent classification tasks. To address these leakage risks, privacy-preserving learning, such as Defensive Distillation, has been developed for defending against poisoning attacks; however, these approaches cannot eliminate all security risks [115] at this time.

2.2 Grayscale Images as Transmission Objects

Before explaining about the system for color images, we will explain the distributed system with chaotic oscillations and synchronization systems using grayscale image as transmitted objects, and numerically test it to obtain the encoded images and the restored images, and compare them with the original images, thereby investigating the feasibility of the proposed approach. Since grayscale images do not involve RGB, the system becomes simpler and hence it is somehow easier to understand than that for color images.

2.2.1 Wave equation with Van der Pol boundary conditions

The system which we consider here is a linear PDE but with a nonlinear boundary condition, which is from the van der Pol equation without forcing

$$\begin{cases} \ddot{x} + (-\alpha\dot{x} + \beta\dot{x}^3) + \omega_0^2 x = 0, \quad \alpha, \beta, \omega^2 > 0 \\ x(0) = x_0, \quad \dot{x}(0) = x_1, \quad x_0, x_1 \in \mathbb{R}, \end{cases} \tag{2.2}$$

where ω_0 is the fixed frequency of the corresponding linear harmonic oscillator. The energy of this system is given by

$$E(t) = \frac{1}{2}[\dot{x}(t)^2 + \omega_0^2 x(t)^2] \tag{2.3}$$

and the time rate of change of energy is

$$\frac{\mathrm{d}}{\mathrm{d}t}E(t) = \dot{x}(t)[\ddot{x}(t) + \omega_0^2 x(t)]$$

$$= \alpha \dot{x}(t)^2 - \beta \dot{x}^4(t), \tag{2.4}$$

so we get

$$\begin{cases} \geq 0, & \text{if} \quad |\dot{x}| \leq \left(\dfrac{\alpha}{\beta}\right)^{\frac{1}{2}} \\ \leq 0, & \text{if} \quad |\dot{x}| > \left(\dfrac{\alpha}{\beta}\right)^{\frac{1}{2}}, \end{cases} \tag{2.5}$$

which shows the self-regulation effect, for the energy will increase when $|\dot{x}|$ is small, and the energy will decrease when $|\dot{x}|$ is large. Therefore, unless the initial condition satisfies $x_0 = x_1 = 0$ in (2.2), casuing $E(t) = 0$ for all $t > 0$, we can know that $E(t)$ will rise and fall between a certain interval of the (B_1, B_2). The bounds B_1 and B_2 can be determined by the parameters α and β.

Let us describe a wave equation below as

$$\frac{1}{c^2} y_{tt}(x,t) - y_{xx}(x,t) = 0, \qquad 0 < x < 1, \quad t > 0, \tag{2.6}$$

which defines the linear PDE that describes a vibrating string on the unit interval $(0, 1)$, where $c > 0$ denotes the speed of wave propagation. Because the speed of wave c is not an essential parameter, we set $c = 1$. Thus, in this study, we consider

$$y_{tt}(x,t) - y_{xx}(x,t) = 0, \qquad 0 < x < 1, \quad t > 0, \tag{2.7}$$

at the left-end $x = 0$, we choose the following boundary condition

$$y_t(0,t) = -\eta y_x(0,t), \qquad t > 0, \quad \eta > 0, \quad \eta \neq 1. \tag{2.8}$$

At the right-end $x = 1$, we assume a nonlinear boundary condition

$$y_x(1,t) = \alpha y_t(1,t) - \beta y_t^3(1,t), \qquad t > 0, \quad \alpha, \beta > 0. \tag{2.9}$$

The initial conditions are

$$y(x,0) = y_0(x), \quad y_t(x,0) = y_1(x), \qquad 0 \leq x \leq 1 \tag{2.10}$$

for two given smooth functions y_0 and y_1.

As a summary, the wave equation with the van der Pol boundary condition is given by

$$\begin{cases} y_{tt}(x,t) - y_{xx}(x,t) = 0, \quad 0 < x < 1, \quad t > 0 \\ y_x(1,t) = \alpha y_t(1,t) - \beta y_t^3(1,t), \quad t > 0 \\ y_t(0,t) = -\eta y_x(0,t), \quad t > 0 \\ y(x,0) = y_0(x), \quad y_t(x,0) = y_1(x), \quad 0 \le x \le 1, \end{cases} \tag{2.11}$$

where we set $\alpha, \beta, \eta > 0$, and $\eta \neq 1$. We use the method of characteristics to rewrite (2.11). Let u and v be the Riemann invariants [27] of (2.11)

$$\begin{cases} u(x,t) = \frac{1}{2}\{y_x(x,t) + y_t(x,t)\} \\ v(x,t) = \frac{1}{2}\{y_x(x,t) - y_t(x,t)\}, \end{cases} \tag{2.12}$$

with initial conditions

$$\begin{cases} u(x,0) = u_0(x) = \frac{1}{2}[y_0'(x) + y_1(x)] \\ v(x,0) = v_0(x) = \frac{1}{2}[y_0'(x) - y_1(x)] \end{cases} \quad 0 \le x \le 1. \tag{2.13}$$

Then substitution of u and v into the boundary condition (2.9) gives

$$u(1,t) = F_{\alpha,\beta}(v(1,t)), \quad t > 0. \tag{2.14}$$

Besides, since $y_x(1,t) = u(1,t) + v(1,t)$, $y_t(1,t) = u(1,t) - v(1,t)$ at the right end $x = 1$, the relationship (2.14) becomes

$$\beta(u-v)^3 + (1-\alpha)(u-v) + 2v = 0. \tag{2.15}$$

Thus, let us summarize what the 1st order hyperbolic equation of (2.13) looks like after using Riemann invariants u and v

$$\Sigma_0 : \begin{cases} u_t(x,t) = u_x(x,t), \quad x \in (0,1), \quad t > 0 \\ v_t(x,t) = -v_x(x,t), \quad x \in (0,1), \quad t > 0 \\ u(1,t) = F_{\alpha,\beta}(v(1,t)), \quad t > 0 \\ v(0,t) = qu(0,t), \quad t > 0 \\ u(x,0) = \frac{1}{2}[y_0'(x) + y_1(x)] = u_0(x), \quad x \in [0,1] \\ v(x,0) = \frac{1}{2}[y_0'(x) - y_1(x)] = v_0(x), \quad x \in [0,1], \end{cases} \tag{2.16}$$

where $q = \frac{1+\eta}{1-\eta}$ by (2.12), from which follows

$$v(0,t) = q(u(0,t)) = \frac{1+\eta}{1-\eta} u(0,t), \quad t > 0. \tag{2.17}$$

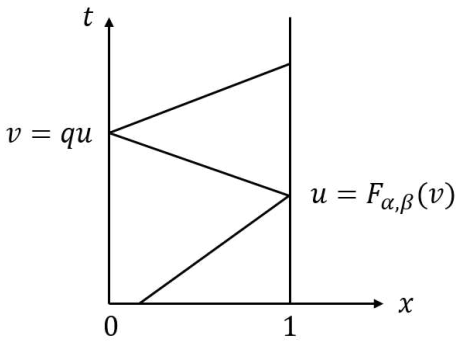


Fig.2.3 Reflection of the characteristic lines. The waves are reflected at $x = 0$ and $x = 1$, where the two functions $v = qu$ and $u = F_{\alpha,\beta}(v)$ are applied.

(2.15) and (2.17) represent the boundary conditions of the two boundaries at which the waves are reflected, respectively, as shown in the Fig. 2.3. When the wave reaches the right end $x = 1$, it is reflected to the left via (2.15) changing direction; when it reaches the left end, it is transmitted to the right via (2.17).

For (2.15), with $0 < \alpha \leq 1$, there exists a unique $u \in \mathbb{R}$ corresponding to each $v \in \mathbb{R}$ [13]. On the other hand, when $\alpha > 1$, for each $v \in \mathbb{R}$, in general there many exist two or three distinct u's $\in \mathbb{R}$ satisfying (2.15). Thus in the latter case, $u = F_{\alpha,\beta}(v)$ is not well-defined, and hence the original PDE system (2.11) is not unique.

For parameters, for example, $\alpha = 0.5, \beta = 1$ and $\eta = 0.625$ are used, but special attention should be paid to the setting of η, which has been shown theoretically to increase exponentially the total variation of the system if it is chosen as an appropriate value. Since total variation is one of the indicators of the severity of the function change, its increase means that the system behaves chaotically.

So far, we have obtained a system of PDEs, which is a wave equation that can cause chaotic oscillations. This system allows us to chaoticize the transmitted image, and in order to restore the chaotic image in a later step, we need to construct a synchronization system. This system is designed so that the system exhibits exactly the same chaotic oscillations of the original system. In the next section we will construct such a synchronization system.

2.2.2 Synchronization system

The construction of a synchronization system requires a portion of the original system's information. Hence, the sender sends two signals to the receiver: one is a secret, coded image, and the other is the signal needed for the synchronized system to restore the original system's state. For the system $\sum_0$ (2.16), consider the following system $\sum_1$:

$$\Sigma_1 : \begin{cases} \hat{u}_t(x,t) = \hat{u}_x(x,t), \quad x \in (0,1), \quad t > 0 \\ \hat{v}_t(x,t) = -\hat{v}_x(x,t), \quad x \in (0,1), \quad t > 0 \\ \hat{u}(1,t) = F_{\alpha,\beta}(v(1,t)), \quad t > 0 \\ \hat{v}(0,t) = qu(0,t), \quad t > 0 \\ \hat{u}(x,0) = \hat{u}_0(x), \quad x \in [0,1] \\ \hat{v}(x,0) = \hat{v}_0(x), \quad x \in [0,1], \end{cases} \tag{2.18}$$

with two signals, $u(0,t), v(1,t)$, as inputs. The relationship between the two systems is shown in Fig. 2.4. We set variables $\widetilde{u} = u - \hat{u}$, $\widetilde{v} = v - \hat{v}$ to obtain the error system as follows:

$$\begin{cases} \widetilde{u}_t(x,t) = \widetilde{u}_x(x,t), \quad x \in (0,1), \quad t > 0 \\ \widetilde{v}_t(x,t) = -\widetilde{v}_x(x,t), \quad x \in (0,1), \quad t > 0 \\ \widetilde{u}(1,t) = 0, \quad t > 0 \\ \widetilde{v}(0,t) = 0, \quad t > 0 \\ \widetilde{u}(x,0) = \widetilde{u}_0(x), \quad x \in [0,1] \\ \widetilde{v}(x,0) = \widetilde{v}_0(x), \quad x \in [0,1]. \end{cases} \tag{2.19}$$

Solving the system using the method of characteristics, we can see that, at any initial value $\widetilde{u}_0$, $\widetilde{v}_0$, the solution $\widetilde{u}(\cdot,t)$ and $\widetilde{v}(\cdot,t)$ is completely zero at moment $t = 2$. In other words, at moment $t = 0$, there is no error between system (2.18) and system (2.16) and they reach a synchronized state.

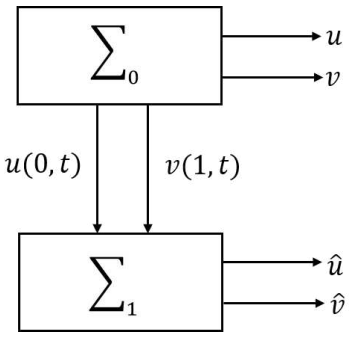


Fig.2.4 Synchronization system $\sum_1$

In the previous study [101], it was assumed that the parameters of the system and information about the boundary conditions necessary to define (2.18) are shared in advance. The sender then sends the modulated information using the states of the chaotic system for secret communication, while the receiver sends the values $u(0,t)$ and $v(1,t)$ at the boundary required for synchronization at the sender to the synchronization system to evolve and demodulate the information by using the system state used during modulation.

2.2.3 Proposed secret communication system

In this study, we use neural networks to improve the security of the method by black-boxing the boundary conditions. Since the experimental objects in this section are grayscale images, the images can be transmitted as a single signal through the secure communication system. In the paper [101], in particular, the modulation and demodulation of the $(M+1)\times(L+1)$ pixel image data are studied by extending the method of the paper [102] to the distribution system. For this purpose, system (2.16) and synchronization system (2.18) must be discretized in both spatial and temporal directions.

Divide the interval $[0,1]$ equally into L parts and set the division points $x_i = i\Delta x(i = 0,1,...,L)$. Here, the interval is given by $\Delta x = \frac{1}{L}$. The time step size is set to Δt and we write $t_k = k\Delta t(k = 0,1,...)$. For the system (2.16) $u(x,t), v(x,t)$ denote the states, $u_i(k), v_i(k)$ denote the approximation values of $u(x_i,t_k), v(x_i,t_k)$ on the

grid points (x_i,t_k). For simplicity, we introduce the $(L+1)$ dimension vectors $u(k) = [u_0(k),u_1(k),...,u_L(k)]^T$, $v(k) = [v_0(k),v_1(k),...,v_L(k)]^T$. Denote $\hat{u}_i(k) =$

$\hat{u}(x_i, t_k)$,
$\hat{v}_i = \hat{v}(x_i, t_k)$ for system (2.18) as well, and denote the $(L+1)$ dimension vector $\hat{u}(k) = [\hat{u}_0(k),\ \hat{u}_1(k), ..., \hat{u}_L(k)]^T$, $\hat{v}(k) = [\hat{v}_0(k), \hat{v}_1(k), ..., \hat{v}_L(k)]^T$. Thus, for example, in Fig. 2.4, the signals $u(0,t), v(1,t)$ sent from $\sum_0$ to $\sum_1$ for synchronization are discretized to $u_0(k), v_L(k)$, respectively. The time evolution of the vector $u(k), v(k)$ is shown in Fig.2.5. We approximate the $\sum_0$ and $\sum_1$ by the upwind difference using the same step sizes in the spatial and temporal directions so that the CFL number is 1. As a result, we obtain an algorithm where $u_i(k)$ is transported to $u_{i-1}(k+1)$ and $v_{i-1}(k)$ is to $v_i(k+1)$. More precisely, the system after discretization is

$$\begin{cases} \frac{u_i(k+1)-u_i(k)}{\Delta t} = \frac{u_{i+1}(k)-u_i(k)}{\Delta x}, & k>0, i \in \{1,2,...,L-1\} \\ \frac{v_i(k+1)-v_i(k)}{\Delta t} = -\frac{v_i(k)-v_{i-1}(k)}{\Delta x}, & k>0, i \in \{1,2,...,L-1\} \\ u_L(k) = F_{\alpha,\beta}(v_L(k)), & k>0, \\ v_0(k) = qu_0(k), & k>0. \end{cases} \tag{2.20}$$

Here, we set $\Delta t = \Delta x = \frac{1}{L}$ as the time and space step sizes, which gives

$$\begin{aligned} u_i(k+1) &= u_{i+1}(k), \quad k>0, i \in \{1,2,...,L-1\} \\ v_i(k+1) &= v_{i-1}(k), \quad k>0, i \in \{1,2,...,L-1\}. \end{aligned} \tag{2.21}$$

$\sum_1$ is discretized in the same manner. Henceforth, the systems $\sum_i (i=0,1)$ after discretization are denoted by $\overline{\sum}_i (i=0,1)$.

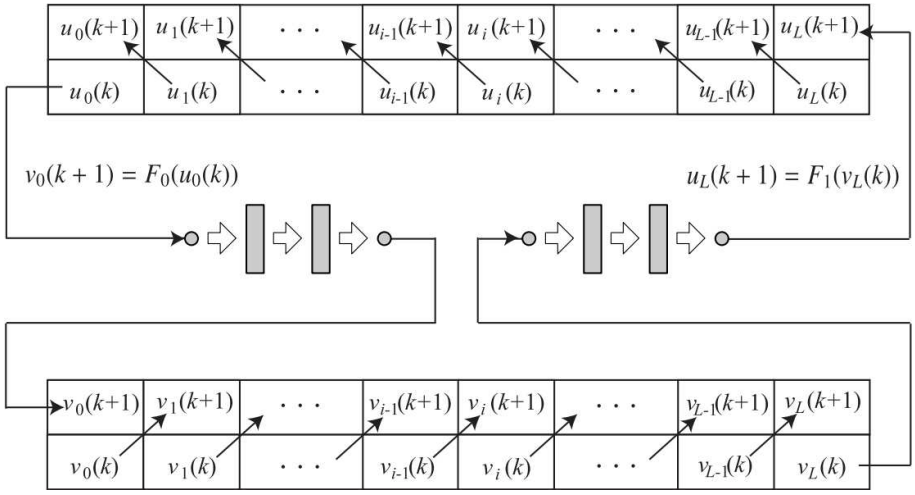


Fig.2.5 Time evolution of $u(k)$ and $v(k)$ in the discretized systems with the boundary conditions given by the neural networks.

As we mentioned earlier, due to the risk of theft due to the small number of parameters of the wave Equation (2.16) with the van der Pol boundary condition, in this study we propose a way to enhance security by approximating the van der Pol boundary condition with a neural network instead of the original boundary condition. The first half of this section describes how the u, v evolve over time, and the two systems $\sum_0, \sum_1$ are discretized. As shown in Fig. 2.5, in the previous method, the waves at the boundaries are updated by $v_0(k+1) = qu_0(k)$ and $u_L(k+1) = F_{\alpha,\beta}(v_L(k))$, respectively. In the proposed method, which uses a neural network instead of these boundary conditions, we need two functions, F_1 and F_0, in order to simulate the boundary conditions.

One of the properties about neural networks is that they can approximate any continuous functions defined on compact sets. That is, neural networks can be a complicated, wiggly function, $f(x)$, as in Fig. 2.6.

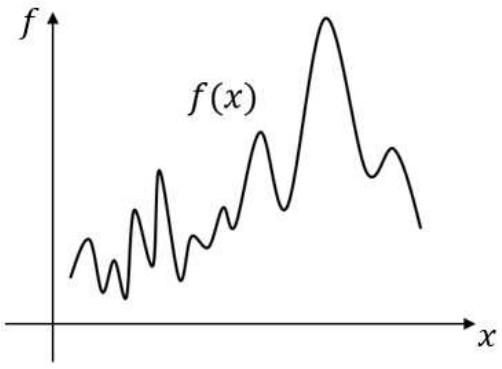


Fig.2.6 A complex continuous function $f(x)$.

For every possible input x, no matter what function it is, there is a neural network whose output value is close to $f(x)$. This property holds for functions with multiple inputs $f = f(x_1, ..., x_m)$ and multiple outputs. This property of neural networks is called the universal approximation property. Moreover, this universality theorem holds even for neural networks that have only one hidden layer between the input and output layers. In other words, the expressive power is extremely high even for extremely simple network structures.

The neural network applied here is composed of seven layers of perceptrons, as shown in Fig. 2.7, and is approximated to $qF_{\alpha,\beta}$ by training. Since in this section

we suppose that the images are of grayscale, both the input and output layers of the neural network have a single neuron. In this proposed approach, the fourth layer is also designed as a single neuron. The whole neural network is divided into two parts; the first part is the first to fourth layers, which is assumed to represent a function F_0, and the second part is the fifth to seventh layers, which is assumed to represent a function F_1. As shown in Fig. 2.5, using the learned neural network, the left boundary condition is replaced, from $v_0(k+1) = qu_0(k)$ to the front half of the neural network F_0, while the right boundary condition is replaced, from $u_L(k+1) = F_{\alpha,\beta}(v_L(k))$ to the back half of the neural network F_1. In this way, the boundary conditions are black-boxed and the information becomes difficult to leak.

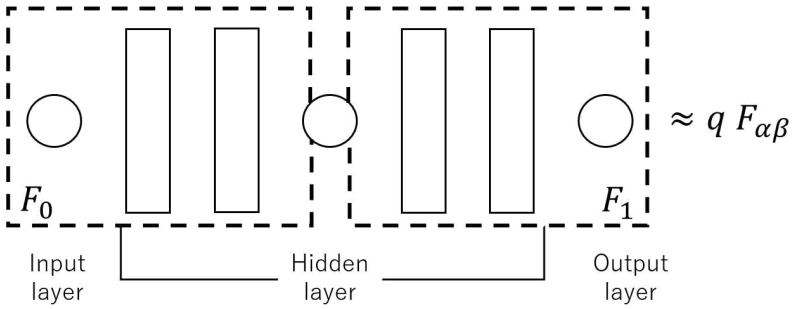


Fig.2.7 Relationship between the structure of neural networks and boundary conditions.

After approximating the van der Pol boundary condition, we tested the resulting new neural network by computing its Lyapunov exponent. In fact, when dealing with actual chaotic phenomena, instead of providing a clear mathematical definition of chaos, whether the system is chaotic or not is checked by a practical condition, which is determined by a certain criterion that is numerically computable. A dynamical system F is chaotic if it satisfies at least one of the following conditions:

(1) F has a sensitive dependence on initial conditions within the defined region.
(2) F has positive Lyapunov exponents at all points in the definite domain excluding the final immobile point [39, 46].

In this study, we use the method of calculating the Lyapunov exponents, that is, condition (2). Lyapunov exponents are used to quantify the separation rate between infinitely close trajectories in a dynamical system. Specifically, under the assumption that linearization is feasible, the separation rate of the two trajectories with an initial

interval of σZ_0 is

$$|\sigma Z(t)| \approx \mathrm{e}^{\lambda t}|\sigma Z_0|, \tag{2.22}$$

where λ is the Lyapunov exponents. In this section, where the images are assumed to be grayscale, since no color issues are involved, it can be viewed as a one-dimensional discrete-time system, in which case the Lyapunov exponent λ for a one-dimensional map $x_{n+1} = L(x_n)$ is defined by

$$\lambda = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N-1} \log |L'(x_i)|, \tag{2.23}$$

where $x \in \mathbb{R}$ is the state variable of the system and $n \in \mathbb{N}$ is the discrete time. If the value of the Lyapunov exponent computed in (2.23) is positive, then the system F is considered to be chaotic. Therefore, we use this method to judge whether the trained neural network successfully approximates the chaotic functions.

The specific components of the whole secret communication system are the system $\sum_0$, synchronization system $\sum_1$, modulation M and demodulation D, where modulation and demodulation, respectively, are represented by the following equations:

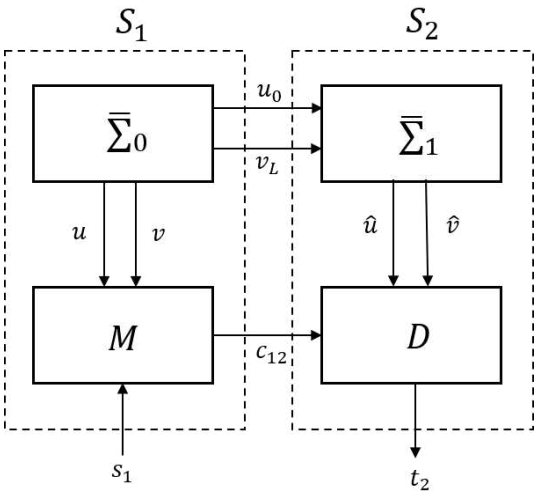


Fig.2.8 Secure communication system.

Modulation

$$M : \begin{cases} w(k+1) = G(u(k), v(k), w(k), s_1(k+1)) \\ c_{12} = w(k), \end{cases} \tag{2.24}$$

Demodulation

$$D : t_2(k+1) = G^{-1}(\hat{u}(k-1), \hat{v}(k-1), c_{12}(k-1), c_{12}(k)), \tag{2.25}$$

where G is the map $G : \mathbb{R}^{L+1} \times \mathbb{R}^{L+1} \times \mathbb{R}^{L+1} \times \mathbb{R}^{L+1} \to \mathbb{R}^{L+1}$ and, for an arbitrarily fixed $a, b, c \in \mathbb{R}^{L+1}$, $G(a, b, c, \cdot)$ is assumed to have an inverse map $G^{-1}(a, b, c, \cdot)$. In addition, the following conditions are also assumed to be satisfied:
(Condition 1:) For any positive number ε, there exists δ such that if

$$\|\xi_1 - \xi_2\|_{\mathbb{R}^{L+1}} < \delta \qquad (\xi_1, \xi_2 \in \mathbb{R}^{L+1}),$$

then the following inequality holds:

$$\sup_{a,b,c \in \mathbb{R}^{L+1}} \|G^{-1}(a, b, c, \xi_1) - G^{-1}(a, b, c, \xi_2)\|_{\mathbb{R}^{L+1}} < \varepsilon.$$

Using the synchronization system $\sum_1$, after the $k = L$ step, $\hat{u}$ and $\hat{v}$ are synchronized to u and v, respectively, so that after the same time, under Condition 1, the restored signal $t_2(k+1)$ is synchronized to the transmitted signal $s_1(k)$. If the original image is encrypted to $(M+1) \times (L+1)$ pixel image data and is sent from subsystem S_1 to subsystem S_2. After the running time needed for the synchronization has passed, the original image data are sliced to $s_1(k) \in \mathbb{R}^{L+1}$ (one row at a time), hence the transmission operation should be performed M times. At the reception side, the image of $(M+1) \times (L+1)$ pixels is obtained by storing the restored data $t_2(k) \in \mathbb{R}^{L+1}$ in sequence.

Remarks. 1. Although the chaotic neural network in the above has the seven layers and is constructed by learning the van der Pol equation, in the proposed method, any neural network can be used as long as it is chaotic. The information required for decryption is the parameters of the neural network representing the boundary conditions. More specifically, the parameters are the matrices and the bias vectors that represent the linear transformations performed in each layer. If the numbers of input and output variables in a certain layer are n_{in} and n_{out}, then the number of parameters in this layer is $n_{\text{in}} \times n_{\text{out}} + n_{\text{out}}$. The sum of this number for each layer is the total number of the parameters, which represents the size of the key space. Since the number of the layers and that of the parameters can be changed freely, the size of the key space can be arbitrarily large. However, the larger the neural network

becomes, the more difficult it is to be trained. Therefore, there is a trade-off between the security performance and the computational complexity of training.

Remarks. 2. The computational cost of the proposed method is as follows. First, the neural network needs to be trained in advance. The time required for this is difficult to estimate because it depends on the quality of the actual data; however, for example, in the following numerical experiments, the average computation time was 1 minute and 48 seconds for the five trials when using Google Colab and Tesla P100. In addition, this pre-training has to be performed once before encryption and decryption. The time required for encryption is the same as the time required to compute a solution of the wave equation. The solution of the wave equation at each position and time can be obtained in a constant time. Hence, the computational cost is proportional to the image size. However, in reality, the computation is much more efficient than this estimation because the computation of the solution to the wave equation can be parallelized in the spatial direction, and the number of processors, especially in GPUs, typically exceeds the numbers of vertical and horizontal pixels in the image. Therefore, actually, encryption can be expected to be possible within a computation time that is several times shorter for the vertical and horizontal sizes of the image. The same is true for decryption.

2.2.4 Numerical experiments

To evaluate the proposed method, numerical experiments were conducted using the neural networks, the training data and the parameters with the following structure.

For the training data in the experiment, the input x was chosen at equal intervals from the interval $(-2, 2)$, and Equation (2.15),

$$\beta(u-v)^3 + (1-\alpha)(u-v) + 2v = 0,$$

was solved using the Newton method, with the target y of the neural network being the solution to the equation. The number of data N_d was set to 20,000. The scatter plot of x and the solutions y for the equation are shown in Fig. 2.9. To reduce the bias of the input data $N = \{x_1, x_2, ..., x_{N_d}\}$ for training, we split the dataset randomly using the train test split function of the Python library Scikit-learn to select the training data and the test data. Here, we use 20% of all data as a test set, so the

number of data in the training set is 16,000 and the number of test sets is 4000. The neural network was implemented using PyTorch, and it was executed on a Tesla K80 on Google Colaboratory.

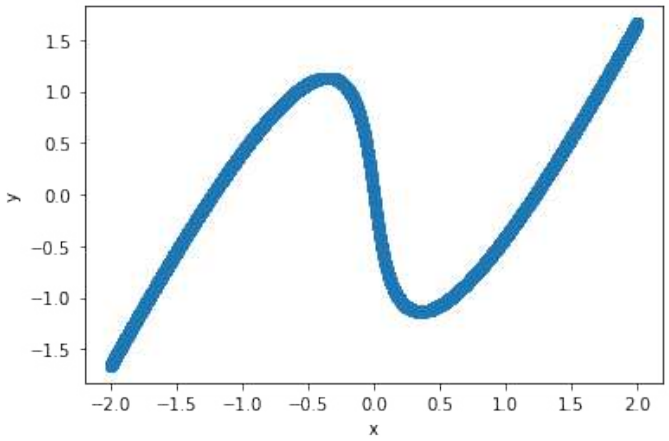


Fig.2.9 Scatter plot of the relationship between x and y.

The neural network used in this study was a seven-layer perceptron with batch normalization layers (the reasons why the batch normalization layers are used are explained later with data in Tables 2.1 and 2.2).

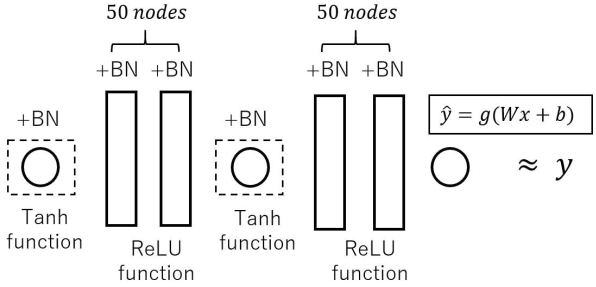


Fig.2.10 Structure of neural networks in numerical experiments.

Each layer has a weight W and a bias b and performs a nonlinear calculation as follows:

$$\hat{y} = g(Wx + b). \tag{2.26}$$

As the training depth of the neural network increases, we apply a batch normalization (BN) to each layer of the neural network. This is to keep the overall distribution

of each layer from biasing towards the upper and lower limits of the interval of the nonlinear function, which leads to the disappearance of the gradient. BN forces the distribution of the input values to follow a standard normal distribution, making the overall learning process more stable. Therefore, each mini-batch for learning should be normalized. For each mini-batch $\mathbb{B} = \{x_1, x_2, ..., x_m\}$, consisting of m data, the average $\mu_{\mathbb{B}}$ and variance $\sigma^2_{\mathbb{B}}$ of the mini-batch are:

$$\mu_{\mathbb{B}} = \frac{1}{N_d} \sum_{i=1}^{N_d} x_i, \tag{2.27}$$

$$\sigma^2_{\mathbb{B}} = \frac{1}{N_d} \sum_{i=1}^{N_d} (x_i - \mu_{\mathbb{B}})^2. \tag{2.28}$$

Batch normalization transforms each data x_i in the mini-batch as follows:

$$\hat{x}_i = \frac{x_i - \mu_{\mathbb{B}}}{\sqrt{\sigma^2_{\mathbb{B}} + \epsilon}}, \tag{2.29}$$

$$y_i = \gamma \hat{x}_i + \beta, \tag{2.30}$$

where γ and β are the parameters of the model so that the output of each layer is normalized. In nonlinear calculations (2.24), $g(\cdot)$ is the activation function. Since we assumed that the images are grayscale, the first and fourth layers are set as neural network layers with only one node. In these layers, the hyperbolic function (tanh),

$$g(r) = \tanh(r) = \frac{\mathrm{e}^r - \mathrm{e}^{-r}}{\mathrm{e}^r + \mathrm{e}^{-r}}, \tag{2.31}$$

is the activation function so that the output converts the input value to a number in the range of -1.0 to 1.0. For layers 2, 3, 5 and 6, 50 nodes were set up and computed using the rectified linear unit (ReLU)

$$g(r) = \begin{cases} 0, & for \quad r < 0 \\ r, & for \quad r \geq 0. \end{cases} \tag{2.32}$$

Training error, which indicates learning accuracy, and test error, which determines generalization performance, were evaluated using the mean square error (MSE),

$$J = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \tag{2.33}$$

where $\hat{y}_i$ is the output value and y_i is the target value. We used Adam as the optimization algorithm for parameter updates in training [55]. The learning rate was set to 0.001 and the number of learning epochs was set to 1000. The training and testing errors, for example, are shown in Fig. 2.11 and Fig. 2.12. Since the training results depend on the random numbers used for parameter initialization, we ran the training ten times while changing the seed of the random numbers, and evaluated the performance of the neural network using the mean and standard deviation of the results. Here, we performed two sets of experiments, one in which each layer of the neural network was nonlinearly transformed using the activation function only, and the other in which batch normalization (refer to BN) layers were added to each layer while performing the nonlinear transformation. The performance results of these two groups are shown in Tables 2.1 and 2.2, respectively.

Table2.1 Mean ± standard deviation of results performed 10 times without BN.

Epoch	**Data Set**	**Mean ± Standard**
1000 times	train set	0.000005 ± 0.000001
	test set	0.000005 ± 0.000001
Lyapunov exponent		−0.067282 ± 0.102745

Table2.2 Mean ± standard deviation of results performed 10 times with BN.

Epoch	**Data Set**	**Mean ± Standard**
1000 times	train set	0.000002 ± 0.000001
	test set	0.000275 ± 0.000394
Lyapunov exponent		0.024377 ± 0.148675

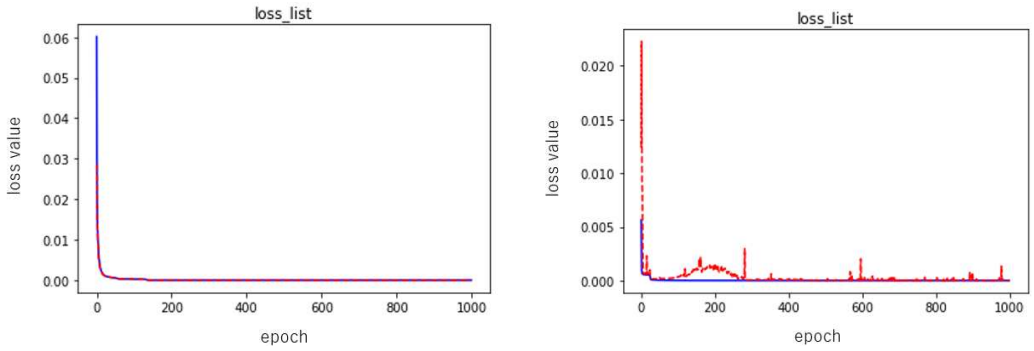


Fig.2.11 Examples of training error (blue line) and testing error (red line) of the neural network during the learning process. The figure on the left is the result of error training without using BN; the figure on the right is the result of error training using BN.

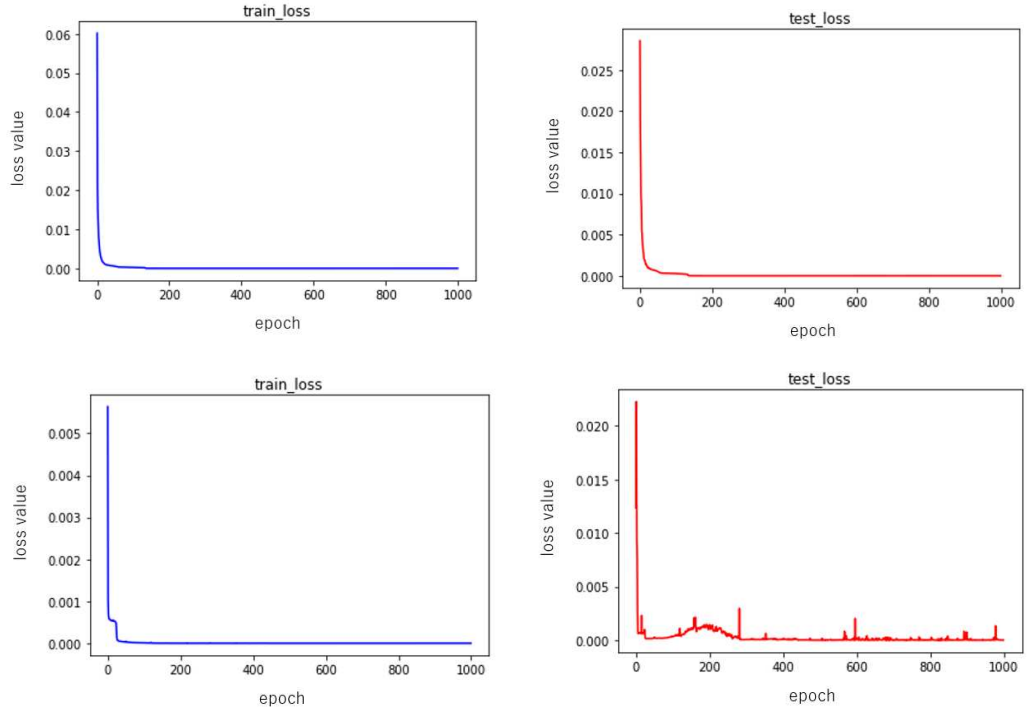


Fig.2.12 Examples of training error drop and test error drop.The 1st figure is training error without BN; the 2nd figure is testing error without BN; the 3rd figure is training error with BN; the 4th figure is testing error with BN.

As can be seen from Fig. 2.11, neither the addition of the batch normalization layers nor the absence of the batch normalization layer affects the speed of the descent, and from the given illustration alone, it can also be said that the descent of the

experimental group without the batch normalization layers is slightly better. Then, in terms of the values used for evaluation, the training and testing errors of the experimental group without batch normalization layers were 0.000005 ± 0.000001 and 0.000005 ± 0.000001, while the group with batch normalization layers had an error drop of 0.000002 ± 0.000001 and 0.000275 ± 0.000394, so we can say that the neural network with batch normalization layers was slightly better than the one without batch normalization layers. This is one reason why we used batch normalization.

Another important reason is the Lyapunov exponent. Introduced at the beginning of this section, the Lyapunov exponent is a numerical value used to judge whether a system is chaotic. Therefore, we also computed the corresponding Lyapunov exponent for the ten experiments, also taking the mean and standard deviation values for the numerical evaluation. From the results presented in Table 2.2, the average value of the Lyapunov exponent for the neural network without batch normalization layers is -0.067282, which is less than 0, indicating that this set of trained neural nets does not approximate the chaotic vibrations well. However, the average value of the Lyapunov exponent for the neural network containing the batch normalization layers is 0.024377, which is greater than 0. From this point of view, it can be said that the neural network successfully approximates the boundary conditions, thereby equipping the chaotic behaviors, and hence being suitable for the secret communication system.

We applied the proposed secret communication system to the image shown in Fig. 2.13 with size 512×512 pixels, i.e. $M = 512, L = 512$.

Modulation

$$\begin{aligned} w(k+1) &= C(w(k))\{0.03m|u(k)|_V + 0.03m|v(k)|_V\} + \{0.5C(w(k)) + 0.1I\} \\ &\quad \times \{0.08m|u(k)|_V + 0.08m|v(k)|_V + s_1(k+1)\}, \\ c_{12}(k) &= w(k). \end{aligned} \tag{2.34}$$

Fig.2.13 Original grayscale image.

The detailed formulas for modulation and demodulation are shown below, respectively.

Demodulation

$$\begin{aligned} t_2(k+1) = \frac{1}{m}\{0.5C(c_{12}(k-1)) + 0.1I\}^{-1} \\ \times \{c_{12}(k) - C(c_{12}(k-1)) \times \{0.03m|\hat{u}(k-1)|_V + 0.03m|\hat{v}(k-1)|_V\}] \\ - 0.08|\hat{u}(k-1)|_V - 0.08|\hat{v}(k-1)|_V, \end{aligned} \tag{2.35}$$

where $w(k) \in \mathbb{R}^{L+1}$, $s_1(k) \in \mathbb{R}^{L+1}$, I is the $(L+1)$ order unit matrix, and $m \in \mathbb{R}$ is the parameter. We also denote $C(f) = \mathrm{diag}(|f_0(1-f_0)|, .., |f_L(1-f_L)|)$ for a vector $f = [f_0, f_1, .., f_L]^T$, and $|f|_V = [|f_0|, |f_1|, ..., |f_L|]^T$. s_1 represents the information to be sent. $c_{12} = w$ is the information sent and received between the systems during communication, and t_2 is the information retrieved by demodulation and corresponds to s_1 if the two systems are synchronized.

Fig. 2.14 shows the results of numerical experiments using a system with boundary conditions set by the learned neural network, with the parameters $m = 6$ in the modulation and demodulation sections. The transmitted image is shown in Fig. 2.13. Fig. 2.14 (left) shows the modulated image after passing through the secret communication system, and Fig. 2.14 (right) shows the image recovered by the synchronous system. In addition, $m = 6$ is a parameter value obtained after numerous attempts, and the size of m affects the outcome of the entire modulation and demodulation operations (see Fig. 2.15).

Fig.2.14 Encrypted grayscale images and restored images in the proposed method of this study. The figure on the left is the encrypted grayscale image; the figure on the right is the restored image.

As shown in Fig. 2.13, the original image is almost unrecognizable from the encrypted image. Fig. 2.14 (right) shows the reconstructed image after the demodulation section. There is little distinction between the transmitted image and the reconstructed image.

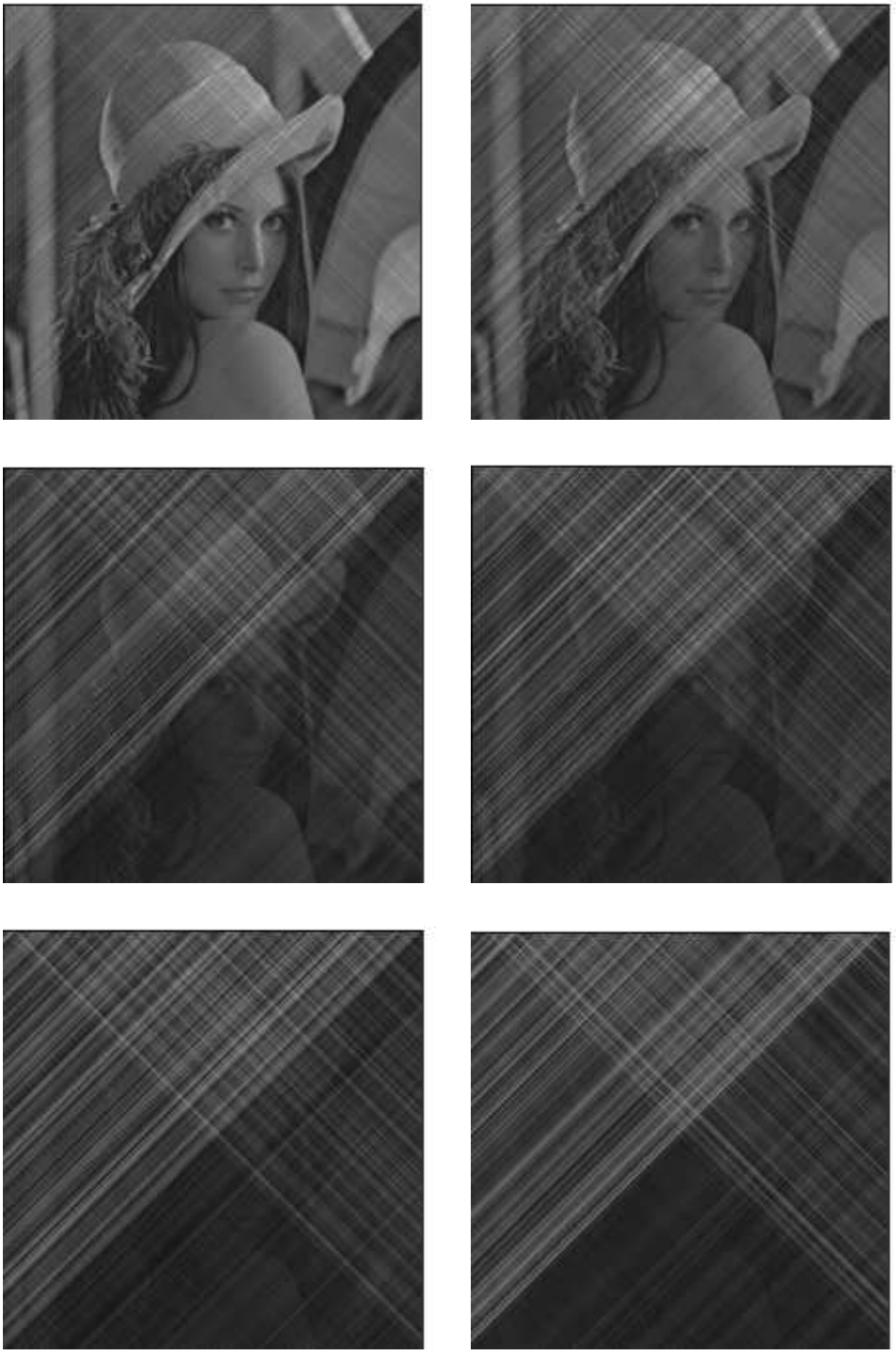


Fig.2.15 The effect of changing the m value in the modulation and demodulation portions on the image encryption effect. Six different values of m (m = 0.8, 1.25, 2.5, 3.0, 4.5, 6.0) were tried, and the corresponding encryption effects were observed for each, where the larger the value of m, the better the secrecy performance of the image, within the computable range.

2.3 Numerical Experiments with Color Images

In the previous section, we explained the wave equations with van der Pol boundary conditions and the corresponding synchronization systems. We then improved the security of the confidential communication system by approximating the chaotic phenomena with a neural network; however, relatively simple grayscale images are used as input objects. Next, we will replace the input images with colored ones and conduct experiments to see if the proposed approach is still applicable to those images. The biggest difference between a color image and a grayscle image is that each pixel of a color image is usually represented by three components, red (R), green (G) and blue (B), of which intensities are represented by numerics between, for example, (0, 255), while a pixel of a grayscale image has a single component. This results in three differences: one being the structure of the neural network itself, the second being that the training method for each pixel is also optional, which means the three components (R, G and B) of each pixel can be trained separately by themselves or mixed together, and the third being the Lyapunov exponent for testing chaotic phenomena.

Color images in the neural network can be computed, one by one, as

$$x_r \to f(x_r) \tag{2.36}$$
$$x_g \to f(x_g) \tag{2.37}$$
$$x_b \to f(x_b). \tag{2.38}$$

The neural network structure in this approach is still one input neuron and one output neuron; they are represented by (2.36)–(2.38) and are trained separately. However, the security of this approach is not very high and there remains a risk of theft. Therefore, in this study, after R, G and B are fed into the neural network, we will mix and disrupt them, adopting a structure that has three input neurons and three output neurons, thus improving the security of the confidential communication system.

The composition diagram of applying RGB to the neural network is shown in Fig. 2.16, where the data from three neurons are used as inputs to the input layer and then sent to the hidden layer . First, since color images are involved, $\sum_0$ and $\sum_1$ correspond to each color in RGB, and the dependent variables u and v are expanded to three dimensions. The u and v are then discretized according to the upwind difference method to obtain the values of u, v at the next time step. (For this part of the process we can refer to Fig. 2.5 in Section 2.2.3). Because the number of inputs

is increased from one to three, the number of the nodes of the first, middle and the last layers are also increased accordingly. By using the input image, the x_r, x_g, and x_b are computed by using chaotic maps as follows:

$$\begin{array}{l} x_r \searrow \\ x_g \to g_i(x_r, x_g, x_b) \to f(g_i(x_r, x_g, x_b)) \quad (i = 1, 2, .., h_n), \\ x_b \nearrow \end{array} \tag{2.39}$$

where h_n is the number of neurons in the hidden layer. The former half of the neural network, F_0, is treated as the left boundary condition, and the latter half, F_1, is treated as the right boundary condition:

$$\begin{aligned} u(1,t) &= F_1(v(1,t)) \\ v(0,t) &= F_0(u(0,t)). \end{aligned} \tag{2.40}$$

We know that the existence of chaos can be determined by calculating the Lyapunov exponent of the dynamical system. If the images are grayscale, the Lyapunov exponent has just one value, hence whether the system is chaotic or not can be determined by examining the exponent. When confronted with color images, the Lyapunov exponent is no longer 1-dimensional due to dimensional growth. k-dimensional space has k Lyapunov exponents. In fact, the Lyapunov exponent $\{\lambda_1, \lambda_2, ..., \lambda_k\}$ is defined as

$$(e, e^{\lambda_1}, e^{\lambda_2}, .., e^{\lambda_k}) = \lim_{N\to\infty}[\text{magnitude of the eigenvalues of } \prod_{n=0}^{N-1} J(x_n)^{1/N}], \tag{2.41}$$

and

$$J(x_n) = (\frac{\partial G_i}{\partial x_i}) \tag{2.42}$$

is the Jacobian matrix of the map $G(x_n)$. Normally, if the maximum Lyapunov exponent λ_1 is positive, the system can be regarded as chaos, but even if $\lambda_1 < 0$, there may be a latent chaotic case which cannot be observed [46]. In this study, as long as one of the Lyapunov exponents is positive, we conclude that the neural network contains chaotic vibrations, which means that the approximation of the chaotic boundary condition is successful.

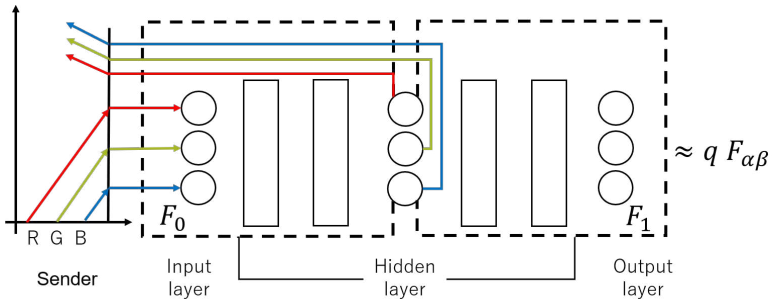


Fig.2.16 Relationship between the structure of Neural Networks and boundary conditions with RGB.

For the numerical experiments, we first trained the neural networks so that the chaotic boundary condition is approximated. We choose training data, from which input data are sampled from $(-2, 2)$. Since a pixel has three components, the total amount of data should preferably be a multiple of 3; we choose $N_d = 30,000$ and (2.15) is again solved by the Newton method. The data are randomly divided into a training set and a test set in the ratio of 8:2. The neural network was implemented using PyTorch, and it was executed on a Tesla T4 from Google Colaboratory.

The structure of the neural network is still a seven-layer multilayer perceptron with three neurons in the first, fourth and seventh layers, and 50 neurons in the other hidden layers. Each layer is nonlinearly transformed by $\hat{y} = g(Wx + b)$ and accompanied by BN, where the activation functions of the first and fourth layers are tanh functions and the others are ReLU functions. The mean squared error is still used for the error function. The entire training process was updated with the Adam algorithm with a learning rate of 0.001. The training was completed after 1000 training runs.

Table 2.3 shows the numerical evaluation of the training results and the test results of the neural network. A total of ten experiments were conducted, and each time the seed value was changed so that the initial values of the parameters of the neural networks are changed. The mean and standard deviation are computed for the experimental results. We can see that the results of training and testing are 0.000694 ± 0.000035 and 0.000919 ± 0.000033, respectively, and Fig. 2.17 shows the error decrease of one of the trainings, from which we can clearly see that the training error and the testing error both decrease rapidly, and there is almost no error rebound. Then, we calculated the Lyapunov exponent, respectively,

$\lambda_1 = 0.1144 \pm 0.1469, \lambda_2 = 0.0038 \pm 0.0917$ and $\lambda_3 = -0.1717 \pm 0.2333$, where λ_1 and λ_2 are positive, so we can conclude that the neural networks are in fact chaotic, which somehow approximates the van der Pol boundary condition.

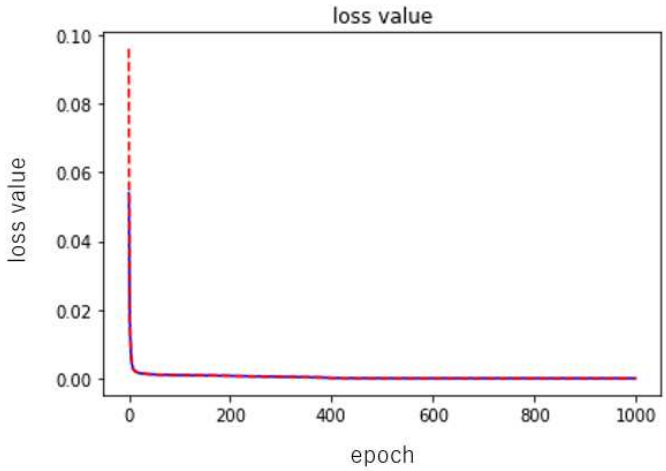


Fig.2.17 Examples of training error (blue line) and testing error (red line) of the neural network during the learning process.

Table2.3 Mean ± standard deviation of results performed at 10 times with RGB.

Epoch	**Data Set**	**Mean ± Standard**
1000 times	train set	0.000694 ± 0.000035
	test set	0.000919 ± 0.000033
Lyapunov exponent		0.1144 ± 0.1469
		0.0038 ± 0.0917
		−0.1717 ± 0.2333

Fig.2.18 Original color image.

Fig.2.19 Encrypted color image and restored image in the proposed method of this study. The figure on the left is the encrypted color image; the figure on the right is the restored color image.

A color image of size 512 × 512, as shown in Fig. 2.19, is put into the whole system as the input. The modulation and demodulation are shown below.

Modulation

$$\begin{aligned} w(k+1) &= C(w(k))\{0.03m|u(k)|_V + 0.03m|v(k)|_V\} + \{0.5C(w(k)) + 0.1I\} \\ &\quad \times \{0.08m|u(k)|_V + 0.08m|v(k)|_V + s_1(k+1)\}, \\ c_{12}(k) &= w(k). \end{aligned} \tag{2.43}$$

Demodulation

$$t_2(k+1) = \frac{1}{m}\{0.5C(c_{12}(k-1)) + 0.1I\}^{-1}$$

$$\times \{c_{12}(k) - C(c_{12}(k-1)) \times \{0.03m|\hat{u}(k-1)|_V + 0.03m|\hat{v}(k-1)|_V\}]$$
$$- 0.08|\hat{u}(k-1)|_V - 0.08|\hat{v}(k-1)|_V. \quad (2.44)$$

With (2.43) and (2.44), we can modulate and recover the images; however, the parameter m still needs to be tried. In Fig. 2.20, we have tried six different values of m. It can be seen that the greater the value of m, the better the secrecy of the image. Fig. 2.18 shows the experimental results when the value of $m = 8.8$. Fig. 2.19 (left) is the image that has been secreted by the proposed system, from which we can conclude that the proposed method is successfully applied to the transmission of color images. In fact, it is very difficult to know the original image from the transmitted one while the restored image shown in Fig. 2.19 (right) is almost identical to the original color image.

It is notable that the method of sharing the neural network as a black box in the whole information transfer system is adopted in this study as the common key cryptosystem. It is well known that the encryption method is based on 2 types of cryptograms: the common key cryptogram and the open key cryptogram. In this context, common key cryptography requires generating a common key for each communication connection, and the key exchange must be secure to prevent being stolen. And as an alternative method of open key cryptogram, using different keys for encryption and decryption. The receiver generates a public key and a private key that is not disclosed to anyone. Thus, it can be seen that as this study transmits the neural network under a common key, it also carries a risk of theft in practice. In order to improve this shortcoming, using open key cryptography to share the neural network will be the topics of my future research.

Fig.2.20 The effect of changing the m value in the modulation and demodulation portions on the image encryption effect. Six different values of m (m = 1.0, 2.0, 5.0, 6.0, 7.5, 8.8) were tried, and the corresponding encryption effects were observed for each, where the larger the value of m, the better the secrecy performance of the image, within the computable range.

2.4 Security Evaluation of the Encrypted Images by the Proposed Method

In the previous section, we numerically examined the proposed method in which the chaotic synchronization system and the artificial neural network are combined. In particular, we tested different values of m to observe the efficiency of the encryption and decryption of the images. In this section, we test the security of the encrypted images. For some different images and the values of m, we investigate whether the encrypted images are secure or not by computing several numerical security measures and color histograms.

Based on traditional evaluation ideas, we first test the randomness of encrypted images: the correlation coefficient and the UACI value [90]. We compare the proposed method with Advanced Encryption Standard (AES) as a reference method. AES is a traditional encryption method, which is a group cipher, where the plaintext is divided into groups of equal length and each group is encrypted until the entire plaintext is encrypted. We used AES in the ECB mode.

Although these are standard measures of security, note that the above measures are not suitable for the proposed approach. In fact, in order to choose suitable measures, the term of "ideal encrypted image" needs to be specified [125]. In typical statistical measures, encrypted images are considered to be secure for, for example, differential attacks when it exhibits randomness. On the other hand, the proposed approach is not designed to generate pseudo-random sequences because a different criterion is used for the term "ideal encrypted image". The proposed approach aims to encrypt the images into almost identical wave images, which are considered to be "ideal" in this study. The idea behind this is that because, for example, the differential attacks essentially try to find how differences in the original images affect the encrypted images, if the encrypted images are almost identical, then the attacks should fail and the information on the original image cannot be retrieved from them.

Although the statistical measures are not necessarily suitable for the proposed method, to a certain extent, the method has a good statistical property as shown below. In addition, we also performed more appropriate tests, in which the distinguishability of two encrypted images is checked.

2.4.1 Randomness testing of encrypted images

Firstly, we computed the correlation coefficients between adjacent pixels for the encrypted image in three different directions: horizontal, vertical and diagonal. Suppose that the image has $N \times N$ pixels $(x_i, y_j),\ i = 1, \ldots, N,\ j = 1, \ldots, N$. We calculate the correlation coefficients for R, G and B values, respectively:

$$r_{xy} = \frac{\frac{1}{N}\sum_{i=1}^{N}\sum_{j}^{N}(x_i - E(x)) \times (y_i - E(y))}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2}\sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_j - E(y))^2}}, \tag{2.45}$$

where $E(x) = \frac{1}{N}\sum_{i}^{N} x_i, E(y) = \frac{1}{N}\sum_{j}^{N} y_j$. The correlation coefficients are between -1 and 1. For example, when the value is close to 1, it means a high correlation between pixels. When it is close to 0, there is no correlation; hence the image is considered pseudo-random. The results are shown in Table 2.4 for the grayscale images and in Table 2.5 for the color images. The images encrypted by the proposed method all present a high inter-pixel correlation, while the traditional AES encryption has an ideal value of around 0. This is because the encrypted images of the proposed method have a certain regularity that is caused by the fact that the images represent the trajectories of waves. Note that this regularity does not imply the recognizability of the original images.

Secondly, we computed UACI values, which measure how much the encrypted image differs from the original image. For two different images, I_1 and I_2, of the same size $N \times N$, the UACI values between them is defined by:

$$UACI = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{I_1(i,j) - I_2(i,j)}{\text{tonal range}} \times 100\%. \tag{2.46}$$

Considering the color range and gray value distribution of images, the ideal value is found to be 33.3. The closer the UACI of the encrypted image is to this ideal value, the better the security.

The computed values are shown in Table 2.4 for the grayscale images and in Table 2.6 for the color images. All the results for AES are about 50.0, while those for the proposed method depend on the target images. AES performs better for the images of vegetables; however, the values of the proposed method for the images of the woman are better than those of AES.

Thirdly, we observed the histograms, which are graphical representations of the intensity distribution of pixels in an image. For gray images, the grayscale histogram reflects the grayscale statistical information of the image. For example, each grayscale image of the previous numerical tests has 256 intensity levels with values from 0 to 255. We store the number of pixels corresponding to each gray level in a 256 capacity array. Similarly, for color images, we can compute the histograms for each of the three different channels, R, G and B, respectively.

The results are shown in Fig. 2.21 for the grayscale images and in Fig. 2.22 for the color images. We can see that the histograms of the images encrypted by AES show a uniform distribution, both in color and grayscale, which to some extent indicates that the encrypted images are disordered and hence in a secure state.

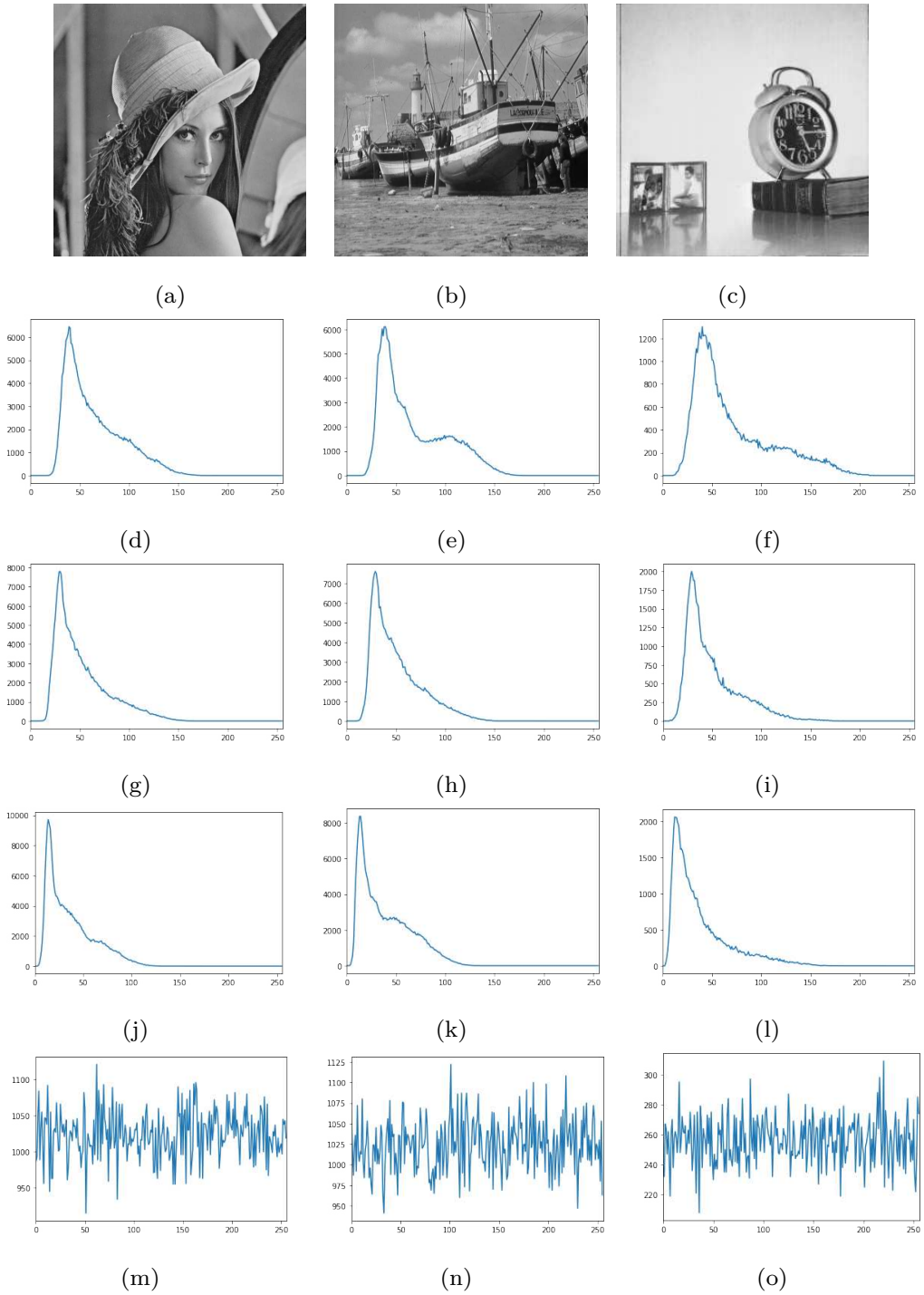


Fig.2.21 The grayscale images and the histograms of the encrypted images. The first row shows three different grayscale images (**a**–**c**). In the second, third and fourth rows, the three figures in each row represent the histograms of the grayscale values of the three original images encrypted by the proposed approach under the settings of $T = 1, 2, 3$ and $m = 6.0$, respectively (**d**–**l**). The last row shows the histogram after encryption by the AES. The horizontal coordinate represents the tonal range and the vertical coordinate represents the absolute frequency (**m**–**o**).

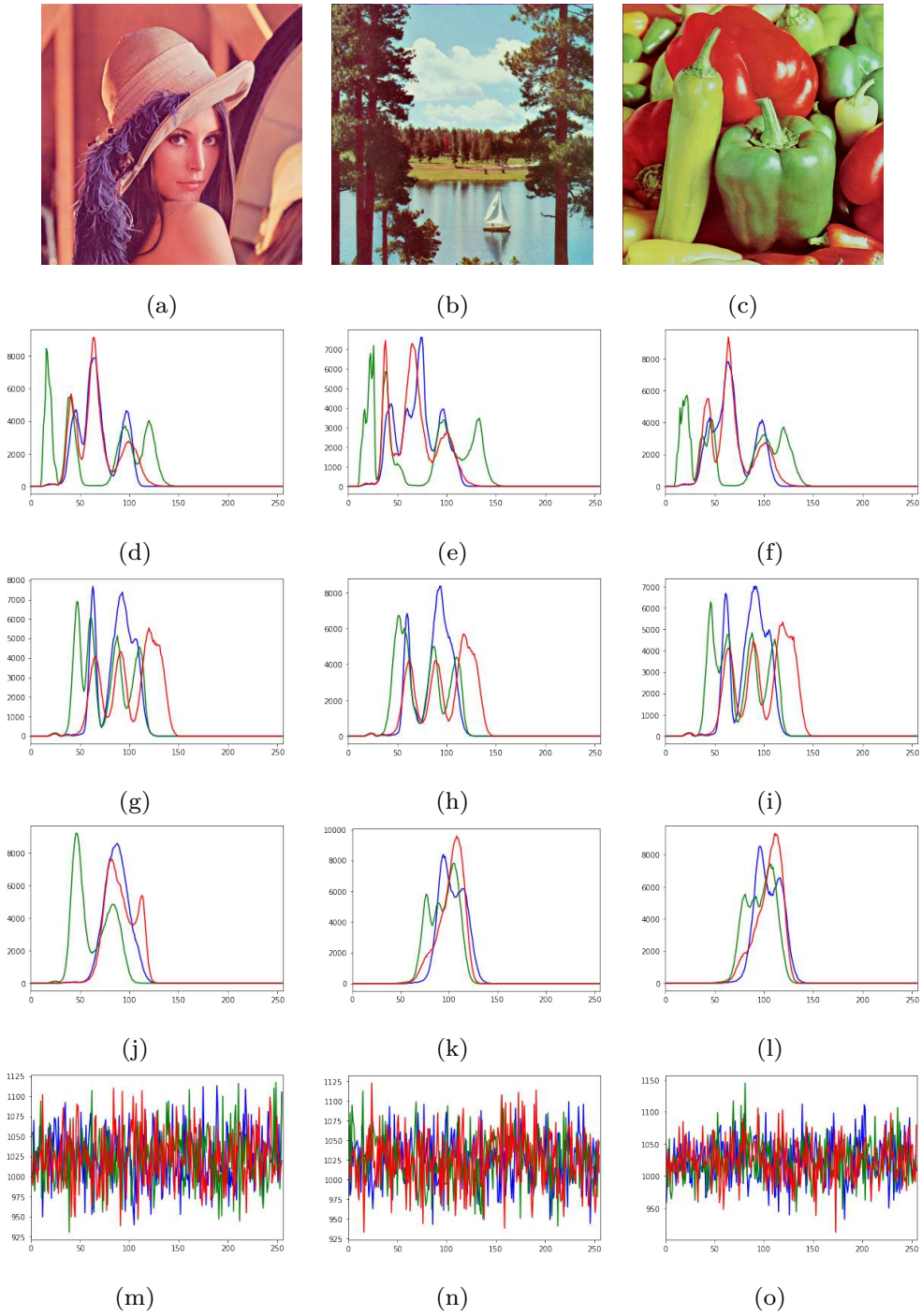


Fig.2.22 The color images and the histogram of the encrypted image. The first row shows three different color images (**a**–**c**). In the second, third and fourth rows, the three figures in each row represent the histograms of the three original images encrypted by the proposed approach under the settings of $T = 1, 2, 3$ and $m = 6.0$, respectively. The last row shows the histogram after encryption by the AES. The red, green and blue colors correspond to the histograms of each channel of R, G and B, respectively (**d**–**l**). The last row shows the histograms after encryption by the AES. The horizontal coordinates represent the tonal range and the vertical coordinates represent the absolute frequency (**m**–**o**).

Table2.4 Correlation test in horizontal, vertical and diagonal directions and UACI test for different grayscale images. Different m ($m = 6, 7.5$) and T ($T = 1, 2, 3$) were tried respectively, where "lena" is Fig. 2.21a, "boat" is Fig. 2.21b, and "clock" is Fig. 2.21c.

Object			**Correlation**			**UACI**
			Horizontal	**Vertical**	**Diagonal**	
lena	m = 6	T = 1	0.972642	0.972993	0.922492	36.141486
		T = 2	0.962627	0.962568	0.890740	39.410799
		T = 3	0.968880	0.969089	0.915038	35.114740
	m = 7.5	T = 1	0.968823	0.969244	0.908995	39.877836
		T = 2	0.954396	0.954471	0.874803	39.730604
		T = 3	0.976311	0.976517	0.940879	35.622222
	AES		0.002630	0.008785	0.000658	49.996347
boat	m = 6	T = 1	0.980538	0.980823	0.943327	36.728787
		T = 2	0.961264	0.961193	0.885866	41.783973
		T = 3	0.973791	0.973994	0.930897	39.928194
	m = 7.5	T = 1	0.972897	0.973300	0.923706	36.761638
		T = 2	0.957479	0.957473	0.874213	41.551513
		T = 3	0.973904	0.974118	0.936444	40.359627
	AES		0.000163	0.000445	0.000502	50.000055
clock	m = 6	T = 1	0.921742	0.922687	0.806450	48.089881
		T = 2	0.858309	0.858372	0.664064	58.067992
		T = 3	0.888560	0.889011	0.722784	58.823428
	m = 7.5	T = 1	0.899274	0.900966	0.770316	47.179087
		T = 2	0.853312	0.957473	0.874213	56.637741
		T = 3	0.880133	0.881703	0.754349	57.388910
	AES		0.005367	0.004363	0.003360	49.929277

Table2.5 Correlation test in horizontal, vertical and diagonal directions with different color images. Different values of m ($m = 7.5, 8.8$) and T ($T = 1, 2, 3$) were tried respectively, where "lena" is Fig. 2.22a, "boat" is Fig. 2.22b, and "veg" is Fig. 2.22c.

Object			**Correlation**								
			Horizontal			**Vertical**			**Diagonal**		
			R	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**
lena	m = 7.5	T = 1	0.97	0.99	0.98	0.97	0.99	0.98	0.95	0.99	0.96
		T = 2	0.99	0.99	0.98	0.99	0.99	0.98	0.98	0.98	0.97
		T = 3	0.93	0.94	0.81	0.93	0.94	0.81	0.82	0.89	0.59
	m = 8.8	T = 1	0.96	0.99	0.97	0.96	0.99	0.97	0.88	0.88	0.88
		T = 2	0.98	0.99	0.98	0.98	0.99	0.97	0.96	0.98	0.96
		T = 3	0.89	0.95	0.82	0.90	0.95	0.83	0.75	0.88	0.57
	AES		5e−3	1e−3	−1e−3	1e-2	6e−3	8e−3	1e−3	−8e−5	3e−3

Table2.5 *Cont.*

Object			Correlation								
			Horizontal			Vertical			Diagonal		
			R	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**
boat	m = 7.5	T = 1	0.96	1.00	0.98	0.97	1.00	0.98	0.94	0.99	0.96
		T = 2	0.99	0.99	0.98	0.99	0.99	0.98	0.98	0.98	0.98
		T = 3	0.92	0.97	0.92	0.93	0.97	0.92	0.82	0.94	0.86
	m = 8.8	T = 1	0.96	0.99	0.97	0.96	0.99	0.97	0.93	0.99	0.95
		T = 2	0.98	0.99	0.97	0.98	0.99	0.97	0.96	0.98	0.96
		T = 3	0.92	0.96	0.92	0.93	0.97	0.93	0.82	0.93	0.87
	AES		6e−4	−1e−3	−1e−4	−3e−3	−1e−3	−3e−3	1e−3	−2e−3	−3e−3
veg	m = 7.5	T = 1	0.97	0.99	0.98	0.97	0.99	0.98	0.94	0.99	0.96
		T = 2	0.99	0.99	0.98	0.99	0.99	0.88	0.98	0.98	0.97
		T = 3	0.92	0.97	0.92	0.93	0.97	0.92	0.82	0.94	0.86
	m = 8.8	T = 1	0.96	0.99	0.97	0.97	0.99	0.98	0.94	0.99	0.96
		T = 2	0.98	0.99	0.97	0.98	0.99	0.97	0.96	0.98	0.96
		T = 3	0.92	0.96	0.92	0.93	0.96	0.92	0.82	0.96	0.86
	AES		2e−4	2e−3	5e−3	−8e−4	2e−3	6e−4	4e−4	4e−3	7e−4

Table2.6 UACI test for different color images. Different values of m ($m = 7.5, 8.8$) and T ($T = 1, 2, 3$) were tried respectively, where "lena" is Fig. 2.22a, "boat" is Fig. 2.22b, and "veg" is Fig. 2.22c.

Object			**UACI**		
			R	**G**	**B**
lena	m = 7.5	T = 1	46.293217	40.515568	24.708355
		T = 2	43.678409	43.262431	37.007906
		T = 3	41.407206	42.583560	29.628579
	m = 8.8	T = 1	46.454520	41.186064	24.949227
		T = 2	44.341227	43.377668	33.920451
		T = 3	43.072341	40.273322	40.672362
	AES		50.133255	50.047464	49.936812
boat	m = 7.5	T = 1	35.143726	48.660561	52.421603
		T = 2	39.571601	55.066602	56.038358
		T = 3	47.716444	53.135069	53.060568
	m = 8.8	T = 1	35.245613	50.609906	55.037498
		T = 2	38.337034	56.456049	58.441035
		T = 3	42.331758	55.305545	56.647502
	AES		49.953939	50.052023	50.175874
veg	m = 7.5	T = 1	37.751291	54.587069	54.226224
		T = 2	45.768913	47.626056	62.656476
		T = 3	37.213291	48.628576	69.999379

Table2.6 *Cont.*

Object		**UACI**		
		R	**G**	**B**
m = 8.8	T = 1	37.915899	54.778044	55.353727
	T = 2	62.202183	48.556424	40.446628
	T = 3	33.298991	51.570749	69.901463
AES		49.956022	50.026667	49.943955

Meanwhile, the histogram of the encrypted image in the mode of the proposed method presents an unbalanced state with a high concentration in a certain region. The reason for this phenomenon is as follows. The proposal method uses the reflection of waves, and even though the initial conditions are random values, they are propagated to the left and right sides along straight lines. In fact, as shown in Fig. 2.23, the images encrypted by AES and the images encrypted by the proposed method are apparently different from each other. The former has almost no structure, and the latter a visible structure that corresponds to the trajectories of the waves. The above mentioned concentration is observed because the intensity of a trajectory takes the same value.

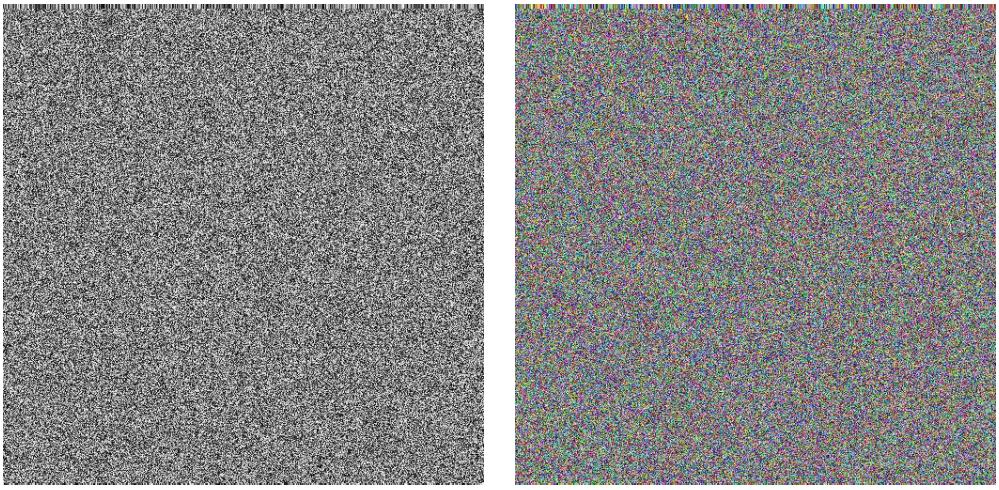


Fig.2.23 The grayscale encrypted image (**left**) and the color encrypted image (**right**) obtained by using the AES on the grayscale image (Fig. 2.13) and the color image (Fig. 2.19).

From this consideration, we deduce that the difference in the position of the highest point of the histogram reflects the difference in the lengths of the straight lines, which in turn depend on the waiting time for synchronization. To confirm this, we computed the histograms changing the waiting time as $T \times$(the vertical size of the target image) with $T = 1, 2, 3$. For grayscale images, the highest point of the histogram increases with the waiting time, and for color images, R, G and B have multiple peaks and the position and the number of the peaks depend on the waiting time. This implies that although the histograms of the encrypted images have some peaks, from these peaks only the waiting time for synchronization can be estimated, and at least in a naive way, no information on the original images can.

2.4.2 Distinguishability of encrypted images

From the above security tests, we can say that AES has higher security from the perspective of randomness; however, the AES and the proposed approach are fundamentally different in the definition of "ideal encrypted images". Hence, it is important to investigate the security issue from another perspective.

Since the ideal encrypted images of the proposed approach are "the indistinguishable images of waves", we calculated the similarity of encrypted images of two different images. If there is a high correlation between the two, it indicates that the proposed

method encrypts different images into almost identical encryption results. In addition, if two encrypted images have high similarity, there is a very low probability of leaking the original image information.

We evaluated three inter-image-similarity criteria [26, 105] for color and grayscale images. Firstly we use the averaging pixel values,

$$\frac{\sqrt{(\overline{I1_R}-\overline{I2_R})^2+(\overline{I1_G}-\overline{I2_G})^2+(\overline{I1_B}-\overline{I2_B})^2}}{\sqrt{3\times(255-0)^2}}, \tag{2.47}$$

where $I1, I2$ are images and the averaging value, for example, $\overline{I_R}$, for the image I is

$$\overline{I_R}=\frac{1}{N_{\text{pixel}}}\sum_{i=0}^{N_{\text{pixel}}} I_{R_i}, I_R=\{I_{R_0}, I_{R_1}, \ldots, I_{R_{N_{\text{pixel}}}}\}.$$

N_{pixel} is the number of pixels and I_{R_i} is the red value of the ith pixel of the image I. Secondly, we use the histogram for pixel values,

$$\frac{\sqrt{\sum_{i=0}^{N_{\text{int}}}(I1_{RH_i}-I2_{RH_i})^2}+\sqrt{\sum_{i=0}^{N_{\text{int}}}(I1_{GH_i}-I2_{GH_i})^2}+\sqrt{\sum_{i=0}^{N_{\text{int}}}(I1_{BH_i}-I2_{BH_i})^2}}{3\times\sqrt{2\times(1-0)^2}},$$
$$I_{RH}=\{I_{RH_0}, I_{RH_1}, \cdots, I_{RH_{N_{\text{int}}}}\}, \tag{2.48}$$

where $I1, I2$ are images and N_{int} is the number of the levels of intensity, and I_{RH_i} is the relative degree of the ith bin of the histogram of the red values of the image I. Thirdly, we use the correlation coefficients:

$$1-\left|\frac{r_{I1I2_R}+r_{I1I2_G}+r_{I1I2_B}}{3}\right|, \tag{2.49}$$

where

$$r_{I1I2_R}=\frac{\sum_{i=0}^{N_{\text{pixel}}}(I1_{R_i}-\overline{I1_R})(I2_{R_i}-\overline{I2_R})}{\sum_{i=0}^{N_{\text{pixel}}}(I1_{R_i}-\overline{I1_R})^2\sum_{i=0}^{N_{\text{pixel}}}(I2_{R_i}-\overline{I2_R})^2}.$$

In the experiments, the resolutions of the two images are unified into the same value. We computed these values changing the value of m and the waiting time for synchronization.

From the results of the grayscale and color images shown in Tables 2.7 and 2.8, it can be seen that the correlation coefficients between the encrypted images of different images are significantly large. The values are indeed almost close to one; when

ten decimal places are used, the values of correlation coefficients are, for example, 0.9999999977 for the grayscale image of "boat" with $m = 7.5, T = 1$ and that of "lena" with $m = 6.0, T = 1$ and 0.9999999865 for the color image of "boat" with $m = 7.5, T = 1$ and that of "lena" with $m = 8.8, T = 2$. The other two measures are very small in all cases, indicating high indistinguishability between the encrypted images, hence the high security of the proposed approach.

Table2.7 Similarity comparison between the grayscale encrypted images $I1$ and $I2$ with the proposed approach. Six different encrypted images are taken, the similarity is calculated between each two, and no comparison is made between encrypted images of the same original image. The result column has three values, from top to bottom, corresponding to (2.47)–(2.49). In particular, "lena" is Fig. 2.21a, "boat" is Fig. 2.21b, and "clock" is Fig. 2.21c.

I1 \ **I2**	**Lena (m = 6, T = 1)**	**Boat (m = 7.5, T = 2)**	**Clock (m = 6, T = 2)**
lena (m = 7.5, T = 1)		0.068396	0.0898650
		0.999999	0.999999
		0.112057	0.119775
boat (m = 7.5, T = 1)	0.042764		0.099446
	0.999999*		0.999999
	0.077088		0.143051
clock (m = 6, T = 3)	0.106253	0.071040	
	0.999999	0.999999	
	0.107990	0.081516	

∗: The value is 0.9999999977 when 10 decimal places are used.

Table2.8 Similarity comparison between the color encrypted images $I1$ and $I2$ with the proposed approach. Six different encrypted images are taken, the similarity is calculated between each two, and no comparison is made between encrypted images of the same original image. The result column has three values, from top to bottom, corresponding to (2.47)–(2.49). In particular, "lena" is Fig. 2.21a, "boat" is Fig. 2.21b, and "veg" is Fig. 2.21c.

I1 \ **I2**	**Lena** **(m = 8.8, T = 2)**	**Boat** **(m = 7.5, T = 2)**	**Veg** **(m = 8.8, T = 1)**
lena (m = 8.8, T = 3)		0.047194 0.999999 0.103503	0.070985 0.999999 0.184824
boat (m = 7.5, T = 1)	0.084080 0.999999* 0.185896		0.008566 0.999999 0.037862
veg (m = 7.5, T = 1)	0.092913 0.999999 0.191360	0.106315 0.999999 0.210313	

∗: The value is 0.9999999865 when 10 decimal places are used.

Chapter 3 Neural Symplectic Form: Learning Hamiltonian Equations on General Coordinate Systems

Hamiltonian mechanics can describe more general equations which are not covered by Lagrangian mechanics, of which the equation of motion is defined as follows:

$$\frac{\partial \mathcal{L}}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}} = 0. \tag{3.1}$$

In the previous models for Hamiltonian equations, this equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} O & I \\ -I & O \end{pmatrix}\begin{pmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{pmatrix} \tag{3.2}$$

is typically assumed, where q is the state variable, and p is a variable called the generalized momentum. For example, the Hamiltonian neural network employs this form of equation. However, the Hamilton equation becomes this form only on special coordinate systems called the Darboux coordinate system [76]. Such coordinate systems often rely on unknown energy functions, thus it is usually not possible to prepare data in this coordinate system. Therefore, it is necessary to build a deep physical model that can be applied to the general coordinate system.

In this chapter, we propose a method for learning Hamilton equations from data represented on general coordinate systems, which are not restricted to generalized momenta. The key ingredient is the neural symplectic form; we proposed to learn the symplectic 2-form by using neural networks from data, thereby learning a coordinate-free representation of Hamiltonian equations. In addition, the Hamilton equation can be represented using a state-dependent skew-symmetric matrix, but not all skew-symmetric matrices are related to the symplectic 2-form. In the proposed method,

in order to restrict the model to symplectic 2-forms, the 1-form that derives the symplectic 2-form is learned by the neural networks. (see Fig. 3.1)

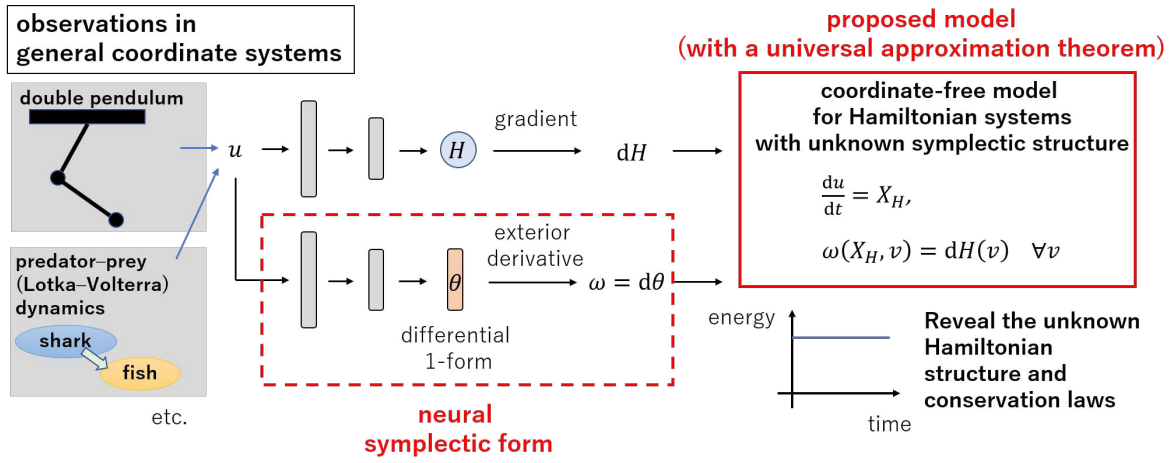


Fig.3.1 Overview of the proposed method. Generally, the analytical representation of generalized momenta is unknown, so the data cannot be presented in a canonical coordinate system. The proposed method learns the Hamilton equation from data represented in an arbitrary coordinate system by learning the symplectic 2-form as well as the energy function. In particular, to ensure that the learned symplectic 2-form is closed, our method learns the differential 1-form that derives the symplectic 2-form. A universal approximation theorem is also provided.

Main contributions of this study include:

1. Symplectic geometric approach to learning symplectic 2-forms. The symplectic 2-form required to describe the Hamilton equation corresponds to a skew-symmetric matrix, but conversely, not all skew-symmetric matrices correspond to a symplectic 2-form. In this chapter, we propose an efficient model by learning the symplectic 1-form that derives the symplectic 2-form with neural networks.

2. Learning the Hamilton equation from data in arbitrary coordinate systems. By using the coordinate-free representation of the Hamilton equation, it is possible to learn the Hamilton equation from data represented in a general coordinate system, not restricted to the Darboux coordinate system. In this way, the proposed method can determine whether the given data can be explained by the hidden theory of classical mechanics or not. A universal approximation theorem is also provided.

This chapter is organized as follows. First, in Section 3.1, we explain existing work related to deep physical models including Lagrangian neural networks and Hamilto-

nian neural networks. We also explain the drawback of each of these models. In Section 3.2, we explain a detailed description of the architecture of our proposed method, the neural symplectic form. In Section 3.3, the numerical experiments are performed using the Hamiltonian neural network, the Lagrangian neural network, the skew matrix learning and the neural symplectic form.

3.1 Physical Models with Neural Networks

Physical models with neural networks can approximate physical phenomena from observations by learning the ordinary differential equation that represents the equation of motion. In order that physical properties are not lost, physical laws such as the law of conservation of energy are to be intrinsically integrated into the model architecture. Such models are being actively explored including the deep physics models based on the Euler–Lagrange equations(3.1) and the Hamiltonian equations (3.2).

Neural Networks for Hamiltonian Mechanics Neural ordinary differential equations (NODE) [15] is a neural network that models the time-derivative of the states, thereby defining an ordinary differential equation in a general way. Due to the generality, this model does not admit the energy conservation law.

Hamiltonian neural network (HNN) is a neural network that models the Hamiltonian H and defines the dynamics following the Hamiltonian mechanics, thereby ensuring the energy conservation law [38]. Although models of the form (3.2) are often used in the previous studies, the Hamilton equation has this form only in the Darboux coordinates [76]. The Darboux coordinate system, which is essentially the generalized momentum p, is defined using the Hamiltonian H, which is the target to be learned. Hence, training data in the coordinate is usually unavailable. In addition, Hamiltonian equations are defined on general symplectic manifolds; however, in the existing studies, cotangent bundles are typically assumed as the symplectic manifold.

A numerical integration of the Hamiltonian system is known to destroy the symplectic structure and does not conserve the Hamiltonian H, unless the integrator is carefully designed (see, e.g., [40, 114]). Several studies focus on the numerical integration that conserves the Hamiltonian H [19, 25, 75, 126, 135]. The Hamiltonian neural network was extended to energy-conserving partial differential equation (PDE) systems, such as the Korteweg–De Vries (KdV) equation [75], to dissipative systems, such as a

Table3.1 Comparison with other studies.

	HNN [38]	LNN [20]	Skew Matrix Learning (Sec. 3)	Neural Symplectic Form (proposed)
In the known Darboux coordinate	yes	yes	yes	yes
In general coordinates on cotangent bundles		yes	yes	yes
On general symplectic manifolds			yes	yes
Only symplectic forms	N/A	N/A		yes

pendulum with friction [75, 134]. In [50], a discrete-time model is proposed for Poisson systems, which are extension of Hamiltonian systems, where the skew symmetric matrix can be degenerate. In particular, in [50], dynamics with state-dependent skew symmetric matrices are leaned by introducing coordinate transformations for learning the dynamics in the latent space. The proposed method is different from this study in that our method does not use the coordinate transformations and hence has an advantage in interpretability. Another approach is employed in [14, 51], where the symplectic map is modeled. SympNets [51] are also shown to be universal approximators, and, in [14], a bound on the prediction error is provided. The proposed method can be combined with these discrete-time approaches.

Modeling using machine learning has also been performed in the field of quantum mechanics, for example, by Tkatchenko and coworkers (e.g. [97, 103]). Some breakthroughs have been reported that have not been possible with conventional computational chemistry methods. The relationship with these studies needs to be investigated in the future.

Neural Networks for Lagrangian Mechanics Another branch of studies focuses on Lagrangian mechanics. Lagrangian neural network (LNN) is a neural network that models the Lagrangian L in a general way [20], and deep Lagrangian network explicitly defines the kinetics energy with a trainable mass matrix [72]. Lagrangian mechanics defines Lagrangian systems on tangent bundles [73], where the state is the pair of the position q and velocity $\dot{q}$. The systems have specific symplectic structures, which are equivalent to Hamiltonian systems in general coordinate systems on cotangent bundles. Because the Lagrangian neural network does not assume equations of a specific form, it can learn a wider class of systems, including a double pendulum, in

addition to the systems that the Hamiltonian neural network can learn (see Table 3.1). Similarly to the Hamiltonian neural network, numerical integrators that preserve the symplectic structure have been investigated [24,98]. Neural network architectures that ensure translational and rotational symmetries have also been investigated [32, 100].

As mentioned before, not all Hamiltonian systems are defined on cotangent bundles. For examples, the some polynomial equations including the Lotka–Volterra equation have a different symplectic structure [42]. In fact, the Lotka–Volterra equation is actually a Hamiltonian system, even though its states are not position, velocity, nor generalized momentum. These equations are out of the scopes of Hamiltonian neural networks and Lagrangian neural networks (see Table 3.1).

3.2 Neural Symplectic Form

In this section, we propose a neural network model based on a coordinate-free representation of Hamiltonian equations. First, we describe this representation. As a precise description of this representation requires detailed geometric knowledge, the relevant details are presented next.

3.2.1 Coordinate-free representation of Hamiltonian equations

In terms of geometry, the Hamilton equation is defined as a flow on a symplectic manifold, which is a pair of a manifold and a symplectic 2-form. Because a flow is defined in a coordinate-free form, the Hamilton equation can be defined in a coordinate-free manner as well. In this section, this is explained in more detail. For further information, see, e.g., [4, 73, 76].

Let $\mathcal{M}$ be a manifold, $T\mathcal{M}$ the tangent bundle and $T^*\mathcal{M}$ the cotangent bundle. For each $q \in \mathcal{M}$, $T_q\mathcal{M}$ denotes the tangent space at q, which is roughly the space of vectors defined locally at q. $T_q^*\mathcal{M}$ is the dual space of $T_q\mathcal{M}$, that is, $T_q^*\mathcal{M}$ is a space of continuous linear maps from $T_q\mathcal{M}$ to $\mathbb{R}$. Differential k-forms are the skew-symmetric multilinear maps from k vectors in $T_q\mathcal{M}$ to $\mathbb{R}$. In particular, a 0-form is a function from $\mathcal{M}$ to $\mathbb{R}$ and a 1-form is a vector in the dual space $T_q^*\mathcal{M}$. Suppose that $\mathcal{M}$ is N-dimensional and has a local coordinate system $x_1, \ldots, x_N$. Typical 1-forms are $\mathrm{d}x_k$'s, each of which maps a vector $v = (v_1, \ldots, v_N)^\top \in T_q\mathcal{M}$ to $\mathrm{d}x_k(v) = v_k \in \mathbb{R}$.

For two 1-forms $\mathrm{d}x_k$ and $\mathrm{d}x_l$, the wedge product $\mathrm{d}x_k \wedge \mathrm{d}x_l$ is a 2-form defined by

$$(\mathrm{d}x_k \wedge \mathrm{d}x_l)(v, w) = v_k w_l - v_l w_k, \quad \text{for all } v = (v_1, \ldots, v_N)^\top, w = (w_1, \ldots, w_N)^\top \in T_q\mathcal{M};$$

in some literature, the wedge product is defined as a constant multiple of the above definition. It follows from the definition that $\mathrm{d}x_k \wedge \mathrm{d}x_l = -\mathrm{d}x_l \wedge \mathrm{d}x_k$, and particularly $\mathrm{d}x_k \wedge \mathrm{d}x_k = 0$.

The exterior derivative d is a linear operator that computes a certain derivative of differential forms. d maps a k-form to a $(k+1)$-form, and has a characteristic property $\mathrm{dd} = 0$. For a 0-form $f(q)$, $\mathrm{d}f$ in the coordinate system $x_1, \ldots, x_N$ is defined by

$$\mathrm{d}f = \frac{\partial f}{\partial x_1}\mathrm{d}x_1 + \cdots + \frac{\partial f}{\partial x_N}\mathrm{d}x_N.$$

For a 1-form $\theta = \sum_{k=1}^N f_k \mathrm{d}x_k$, the exterior derivative is

$$\mathrm{d}\theta = \mathrm{d}\sum_{k=1}^N f_k \mathrm{d}x_k = \sum_{l=1}^N \sum_{k=1}^N \frac{\partial f_k}{\partial x_l}\mathrm{d}x_l \wedge \mathrm{d}x_k = \sum_{l<k} \left(\frac{\partial f_k}{\partial x_l} - \frac{\partial f_l}{\partial x_k}\right) \mathrm{d}x_l \wedge \mathrm{d}x_k.$$

For 2 vectors $v = (v_1, \ldots, v_N)^\top, w = (w_1, \ldots, w_N)^\top \in T_q\mathcal{M}$, the value of $\mathrm{d}\theta$ is represented by using a skew matrix W

$$\begin{aligned}
\mathrm{d}\theta(v, w) &= \sum_{l<k} \left(\frac{\partial f_k}{\partial x_l} - \frac{\partial f_l}{\partial x_k}\right) \mathrm{d}x_l \wedge \mathrm{d}x_k(v, w) \\
&= \sum_{l<k} \left(\frac{\partial f_k}{\partial x_l} - \frac{\partial f_l}{\partial x_k}\right) (v_l w_k - v_k w_l) \\
&= \begin{pmatrix} v_1 v_2 \cdots v_N \end{pmatrix} W \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}, \quad W = \begin{pmatrix} 0 & \frac{\partial f_2}{\partial x_1} - \frac{\partial f_1}{\partial x_2} & \frac{\partial f_3}{\partial x_1} - \frac{\partial f_1}{\partial x_3} & \cdots \\ \frac{\partial f_1}{\partial x_2} - \frac{\partial f_2}{\partial x_1} & 0 & \frac{\partial f_3}{\partial x_2} - \frac{\partial f_2}{\partial x_3} & \cdots \\ \frac{\partial f_1}{\partial x_3} - \frac{\partial f_3}{\partial x_1} & \frac{\partial f_2}{\partial x_3} - \frac{\partial f_3}{\partial x_2} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.
\end{aligned}$$

Differential 2-forms are generally represented by using a skew matrix in the same way.

Definition 3.1. A differential form ω is closed if $\mathrm{d}\omega = 0$.

Definition 3.2. A differential 2-form ω is non-degenerate if the skew matrix associated with ω is non-degenerate.

Definition 3.3. A symplectic 2-form is a closed and non-degenerate differential 2-form.

A Hamiltonian equation on a symplectic manifold $\mathcal{M}$ with the symplectic 2-form ω is defined in a coordinate-free way as

$$\frac{\mathrm{d}u}{\mathrm{d}t} = X_H, \quad \omega(X_H, \cdot) = \mathrm{d}H(\cdot), \tag{3.3}$$

where H is a Hamiltonian and X_H is a vector field defined by the second equality of the above equations. Note that the differential forms are functions of vectors, which are defined regardless of the coordinate systems. Hence, the model (3.3) is a coordinate-free representation.

For example, suppose that $\mathcal{M}$ is a 2-dimensional manifold with a local coordinate $(q, p)^\top$. $\mathrm{d}q \wedge \mathrm{d}p$ is a symplectic 2-form on this manifold; in fact, $\mathrm{d}q \wedge \mathrm{d}p$ is obtained as the exterior derivative of 1-form $\theta = q\mathrm{d}p$:

$$\mathrm{d}\theta = \mathrm{d}(q\mathrm{d}p) = \mathrm{d}q \wedge \mathrm{d}p$$

and hence $\mathrm{d}q \wedge \mathrm{d}p$ is closed:

$$\mathrm{d}(\mathrm{d}q \wedge \mathrm{d}p) = \mathrm{dd}(q\mathrm{d}p) = 0$$

because $\mathrm{dd} = 0$. The matrix representation of this 2-form is

$$(\mathrm{d}q \wedge \mathrm{d}p)(\begin{pmatrix} q_1 \\ p_1 \end{pmatrix}, \begin{pmatrix} q_2 \\ p_2 \end{pmatrix}) = \begin{pmatrix} q_1 & p_1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} q_2 \\ p_2 \end{pmatrix}$$

because

$$(\mathrm{d}q \wedge \mathrm{d}p)(\begin{pmatrix} q_1 \\ p_1 \end{pmatrix}, \begin{pmatrix} q_2 \\ p_2 \end{pmatrix}) = \mathrm{d}q(\begin{pmatrix} q_1 \\ p_1 \end{pmatrix})\mathrm{d}p(\begin{pmatrix} q_2 \\ p_2 \end{pmatrix}) - \mathrm{d}q(\begin{pmatrix} q_2 \\ p_2 \end{pmatrix})\mathrm{d}p(\begin{pmatrix} q_1 \\ p_1 \end{pmatrix}) = q_1 p_2 - q_2 p_1.$$

Meanwhile, $\mathrm{d}H$ is computed as

$$\mathrm{d}H = \frac{\partial H}{\partial q}\mathrm{d}q + \frac{\partial H}{\partial p}\mathrm{d}p.$$

Therefore, by substitution of $\mathrm{d}u/\mathrm{d}t = X_H$ with $u = (q, p)^\top$ into the second equation, the coordinate-free form (3.3) becomes

$$\mathrm{d}q \wedge \mathrm{d}p \left(\frac{\mathrm{d}}{\mathrm{d}t} \begin{pmatrix} q \\ p \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \right) = \mathrm{d}H(\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}) \quad \text{for all} \quad v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

of which matrix representation is

$$\begin{pmatrix} \frac{\mathrm{d}q}{\mathrm{d}t} & \frac{\mathrm{d}p}{\mathrm{d}t} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \frac{\partial H}{\partial q} \mathrm{d}q(\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}) + \frac{\partial H}{\partial p} \mathrm{d}p(\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}) \quad \text{for all} \quad v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}.$$

This is equivalent to

$$v_2 \frac{\mathrm{d}q}{\mathrm{d}t} - v_1 \frac{\mathrm{d}p}{\mathrm{d}t} = \frac{\partial H}{\partial q} v_1 + \frac{\partial H}{\partial p} v_2 \quad \text{for all } v_1, v_2,$$

which gives the standard form of the Hamilton equation:

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial H}{\partial p}, \quad \frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial H}{\partial q}.$$

Based on the above prior knowledge of the fundamentals of geometric mechanics, the model in this chapter can be used on a general symplectic manifold, but for simplicity, we describe the case where the phase space is $\mathcal{M} = \mathbb{R}^{2N}$. A differential 2-form ω on $\mathcal{M}$ is a skew-symmetric bilinear function that maps given two vectors into a real number, depending on each point u on $\mathcal{M}$. The skew-symmetric bilinear function defined by ω has the following matrix representation:

$$\omega_u(v_1, v_2) = v_1^\top W_u v_2, \quad \text{for all } v_1, v_2 \in \mathbb{R}^{2N},$$

where W_u is a skew-symmetric matrix, and the subscript u denotes that ω and hence its matrix representation W_u depend on u. A symplectic 2-form is a differential 2-form that is nondegenerate and closed.

The following is the coordinate-free form of Hamiltonian equations [73]

$$\frac{\mathrm{d}u}{\mathrm{d}t} = X_H, \ \omega(X_H, v) = \mathrm{d}H(v) \quad \text{for all } v \in \mathbb{R}^{2N}. \tag{3.4}$$

Here, $\mathrm{d}H$ is the Fréchet derivative of the Hamiltonian H, and X_H is a vector field depending on H. This equation is satisfied regardless of the coordinate system in which the state variable u is expressed. Therefore, by using this equation as a model, as long as the given data is described by the Hamilton equation, it is possible to learn both the symplectic 2-form and the Hamiltonian that define the Hamilton equation, no matter what coordinate system the data is given in.

3.2.2 De Rham cohomology and the de Rham theorem

From the property $\mathrm{dd} = 0$ of the exterior derivatives, $\mathrm{Im\ d} \subset \mathrm{Ker\ d}$. The difference $\mathrm{Ker\ d}/\mathrm{Im\ d}$ is called the de Rham cohomology space.

As is well known, there is a natural duality between differential k-forms ω's and k-dimensional integral domains Ω's in the sense that a real number can be associated with each pairing of $\langle \omega, \Omega \rangle$ in the following way

$$\langle \omega, \Omega \rangle := \int_\Omega \omega.$$

In particular, the Stokes theorem for differential forms

$$\int_\Omega \mathrm{d}\omega = \int_{\partial\Omega} \omega$$

gives the duality between the exterior derivative d and the boundary operator ∂

$$\langle \mathrm{d}\omega, \Omega \rangle = \langle \omega, \partial\Omega \rangle.$$

This duality associates differential forms, the exterior derivative and the cohomology with integral domains, the boundary operator and the homology, respectively. In fact, the boundary operator ∂ is defined in such a way that ∂ maps k-dimensional domain to its $k-1$-dimensional boundary. Because a boundary of a domain is in general a cycle, the boundary of a boundary is 0: $\partial\partial = 0$. This property of the boundary operators is the same as that of the exterior derivative d, and hence the similar space to the cohomology space can be introduced. In fact, because $\mathrm{Im}\ \partial \subset \mathrm{Ker}\ \partial$, we can consider the difference $\mathrm{Ker}\ \partial/\mathrm{Im}\ \partial$. This space is the homology space. The domain contained in the homology space is essentially a "hole," because basically it is a cycle that is not a boundary of any other domain. The de Rham theorem states that there is an isomorphism between the cohomology space and the homology space. In particular, the dimension of the cohomology space is the same as that of the homology space, the number of holes. Thus if the underlying phase space has no hole, the cohomology space vanishes and $\mathrm{Im\ d} = \mathrm{Ker\ d}$. When the cohomology space does not vanish, the members of this space must be computed and added to the model. This is possible because this space is finite-dimensional, and hence we can enumerate the members.

3.2.3 Naive method: the skew matrix learning

By replacing the symplectic 2-form with the matrix W_u, (3.6) can be rewritten as

$$\frac{\mathrm{d}u}{\mathrm{d}t} = X_H,\ X_H^\top W_u v = v \cdot \nabla H \text{ for all } v \in \mathbb{R}^{2N} \quad \Leftrightarrow \quad \frac{\mathrm{d}u}{\mathrm{d}t} = W_u^{-\top} \nabla H.$$

A natural model based on this representation would be a model in which W_u and H are modeled by multilayer perceptrons:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = W_{u,\mathrm{NN}}^{-\top} \nabla H_{\mathrm{NN}}(u), \tag{3.5}$$

where H_{NN} is a function of u defined by a multilayer perceptron, and $W_{u,\mathrm{NN}}$ is a skew-symmetric matrix depending on u represented by another multilayer perceptron. We refer this model as *the skew matrix learning*. If u is represented in the Darboux coordinate system, (3.5) becomes (3.2), and hence the model is the same as the Hamiltonian neural network.

Similarly to Hamiltonian neural networks, this model has the energy conservation law.

Theorem 3.1. *Solutions to* (3.5) *satisfy* $\mathrm{d}H_{\mathrm{NN}}/\mathrm{d}t = 0$.

Proof. By the chain rule, we have

$$\frac{\mathrm{d}H_{\mathrm{NN}}(u)}{\mathrm{d}t} = \nabla H_{\mathrm{NN}} \cdot \frac{\mathrm{d}u}{\mathrm{d}t}.$$

Substituting (3.5) into the above equation, we get

$$\frac{\mathrm{d}H_{\mathrm{NN}}(u)}{\mathrm{d}t} = \nabla H_{\mathrm{NN}} \cdot W_{u,\mathrm{NN}}^{-\top} \nabla H_{\mathrm{NN}} = 0$$

because $W_{u,\mathrm{NN}}^{-\top}$ is skew-symmetric and for any skew-symmetric matrix M and for any vector v

$$v \cdot Mv = 0,$$

which is confirmed by

$$v \cdot Mv = v^\top Mv = (v^\top Mv)^\top = v^\top M^\top v = -v^\top Mv = -v \cdot Mv$$

because M is skew-symmetric and $v^\top Mv$ is a 1×1 matrix, for which $(v^\top Mv)^\top = v^\top Mv$ holds. The above equality gives $2v \cdot Mv = 0$. □

3.2.4 Neural symplectic form

Although the model in the previous section is natural, it is a redundant model and inefficient for learning as explained below. As explained in the beginning of this chapter, although a differential 2-form corresponds to a skew-symmetric matrix, not all skew-symmetric matrices define a symplectic 2-form. Symplectic 2-forms have the characteristic feature of being closed. In this study, we propose a model in which the learned 2-form is guaranteed to be closed. We refer the learned 2-form as *the neural symplectic form*.

First, the necessary terminology is briefly explained. See Section 3.2.1 for details. A differential 0-form on $\mathcal{M} = \mathbb{R}^{2N}$ is a function from $\mathcal{M}$ to $\mathbb{R}$. A differential 1-form θ on $\mathcal{M} = \mathbb{R}^{2N}$ is a field of linear functions each of which maps a vector $v \in \mathbb{R}^{2N}$ to $\theta_u(v) \in \mathbb{R}$, depending on each point $u \in \mathcal{M}$. In general, a linear function from $\mathbb{R}^{2N}$ to $\mathbb{R}$ can be expressed as an inner product with a vector, so a differential 1-form can be expressed as a vector field depending on u. A differential operation called the exterior derivative d is defined for differential forms. The exterior derivative is a graded linear map, i.e. a linear map depending on an integer k, which transfers a differential k-form to a differential $(k+1)$-form and has the property that $\mathrm{dd} = 0$.

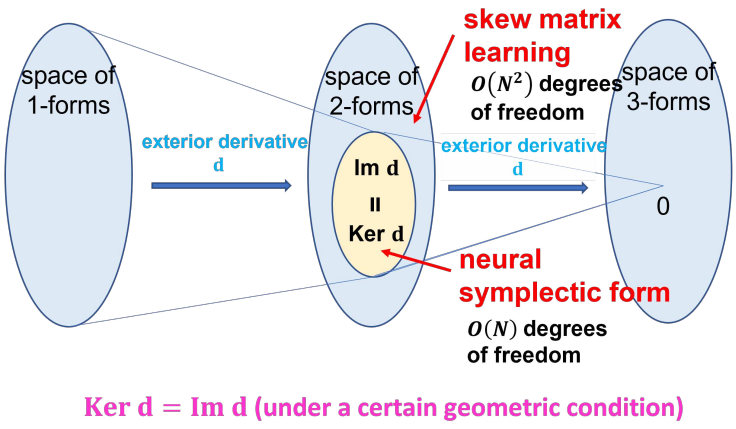


Fig.3.2 The de Rham theorem ensures the Ker d = Im d.

The differential form in Ker d is called a closed form. Since a symplectic 2-form is a closed form, in order to learn the symplectic 2-form by neural networks, neural networks should be designed so that differential 2-forms represented by the neural networks are contained in Ker d. Meanwhile, due to the property of the exterior

derivative, $\mathrm{dd} = 0$, it holds that Im d $\subset$ Ker d.

Actually, according to the de Rham theorem, when the phase space is $\mathbb{R}^{2N}$, these two spaces coincide: Im d = Ker d. The difference between these two spaces Im d/Ker d is called the cohomology space. The de Rham theorem states that the cohomology space is isomorphic to the homology space, which is roughly a space of "spatial holes." Because $\mathbb{R}^{2N}$ contains no holes, the homology space must vanish, and hence Im d = Ker d holds (see Fig. 3.2.) Even when the phase space has a hole, in many cases the space can be embedded in a large Euclid space without holes, and the model should be defined on that space. See Section 3.2.2.

Therefore, in this study, instead of learning the symplectic 2-form directly, we propose a method to learn the symplectic 2-form by learning the differential 1-form of which exterior derivative gives the symplectic 2-form. The following is the coordinate-free form of the proposed model:

$$\tilde{\omega} = \mathrm{d}\theta_{\mathrm{NN}}, \quad \frac{\mathrm{d}u}{\mathrm{d}t} = \tilde{X}_{H_{\mathrm{NN}}}, \quad \omega(\tilde{X}_{H_{\mathrm{NN}}}, v) = \mathrm{d}H_{\mathrm{NN}}(v) \quad \text{for all } v \in \mathbb{R}^{2N} \tag{3.6}$$

Unlike (3.5), in this model skew-symmetric matrices that do not correspond to symplectic forms are not explored. Therefore, this model can be trained much more efficiently than (3.5). In fact, suppose that the dimension of the phase space is $2N$. Then, the number of components of the skew-symmetric matrix is $N(2N-1)$, while the differential 1-form is represented as a vector field, the number of which components is $2N$. Consequently, a model using the neural symplectic form can reduce the order of the number of functions to be learned from $O(N^2)$ to $O(N)$ without sacrificing the expressive power. As per the above, the differential 1-form can be expressed as a vector field. Hence the neural network for modeling the 1-form in the proposed model essentially models a vector field Y_{NN}, which represents the differential 1-form θ_{NN}. As shown in Section 3.2.1, the vector field Y_{NN} is transformed to the matrix $\tilde{W}_u$ representing the symplectic 2-form as follows:

$$(\tilde{W}_u)_{i,j} = \frac{\partial (Y_{\mathrm{NN}})_i}{\partial u_j} - \frac{\partial (Y_{\mathrm{NN}})_j}{\partial u_i}.$$

In the actual model, Y_{NN} given by the neural network is differentiated by the automatic differentiation, and $W_{u,\mathrm{NN}}$ is obtained by substituting the derivatives into the above equation. Thus we have the model expressed in terms of vectors and a matrix,

without using the differential forms:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \tilde{W}_u^{-\top} \nabla H_{\mathrm{NN}}(u), \quad (\tilde{W}_u)_{i,j} = \frac{\partial (Y_{\mathrm{NN}})_i}{\partial u_j} - \frac{\partial (Y_{\mathrm{NN}})_j}{\partial u_i}. \tag{3.7}$$

It is important to note that the neural symplectic form model can be built on any coordinate system. We now have two representations of the neural symplectic form model, the differential form (3.6) and the matrix form (3.7), where the calculation of the matrix W corresponding to the symplectic 2-form ω is dependent on the coordinate system. In fact, to select a coordinate system, bases must be specified and different bases give a different coordinate system. The matrix W is transformed according to the choice of the bases. This means that we can find W on any coordinate system, thus enabling the neural symplectic form model to be modelled from general coordinates.

The proposed model has the energy conservation law and the universal approximation property.

Theorem 3.2. *Solutions to* (3.6)*, or equivalently, to* (3.7) *satisfy* $\mathrm{d}H_{\mathrm{NN}}/\mathrm{d}t = 0$.

Proof. The proof of this theorem is same as the proof of Theorem 3.1. □

We denote by $\Sigma(\sigma)$ the space of the neural networks with the activation function σ:

$$\Sigma(\sigma) = \{g : \mathbb{R}^r \to \mathbb{R} \mid g(x) = \sum_i^q \beta_i \sigma(\gamma_i^\top x + \alpha_i), \alpha_i \in \mathbb{R}, \beta_i \in \mathbb{R}, \gamma_i \in \mathbb{R}^r\}.$$

Theorem 3.3. *Suppose that the Hamilton equation to be learned can be represented in the form* (3.4) *using a Hamiltonian* $H \in W^{1,p}$ *and a symplectic 2-form* ω *that is derived from a 1-form* $\theta \in W^{\mathrm{d},p}$ *in the sense that*

$$\mathrm{d}\theta = \omega.$$

Suppose also that the phase space is compact. The model (3.6) *with neural networks in* $\Sigma(\sigma)$*, of which activation function* σ *is in* C^∞ *and does not vanish everywhere, has the universal approximation property in the sense that* θ *and* H *can be approximated by the neural networks with arbitrary accuracy.*

$W^{1,p}$ and $W^{\mathrm{d},p}$ are Sobolev spaces; see [2, 47].

Proof. In the spaces $W^{1,p}$ and $W^{\mathrm{d},p}$, the spaces of C^∞ functions $C^\infty \cap W^{1,p}$ and $C^\infty \cap W^{\mathrm{d},p}$ are respectively dense [47]. Therefore, if neural networks used in the model admit the universal approximation property to C^∞ functions, then the universal approximation theorem for the differential forms holds. Meanwhile, regarding the approximation of C^∞ functions, the following theorem is known.

Theorem (Hornik et al., 1990)**.** *If the activation function $\sigma \neq 0$ belongs to $S_p^m(\mathbb{R})$ for an integer $m \geq 0$, then $\Sigma(\sigma)$ is m-uniformly dense in $C^\infty(K)$, where K is any compact subset of $\mathbb{R}^N$.*

$S_p^m(\mathbb{R})$ is the Sobolev space, which is roughly functions with up to mth (weak) derivatives with the bounded L^p norm. $W^{1,p}$ and $W^{\mathrm{d},p}$ are Sobolev spaces of differential forms; see [2, 47] for the definitions and the properties. Hence, if the activation function σ of the hidden layer is in $C^\infty \subset S_p^m(\mathbb{R})$ and does not vanish everywhere, then for any C^∞ function, there exists a neural network that approximates this function. Since it is assumed that σ is C^∞ and does not vanish everywhere, we need to prove that σ and its derivatives are in L^p. Because the phase space is assumed to be compact and the activation functions are smooth, the outputs of the neural network and hidden layers are also compact. Hence, the activation function σ is essentially used on the compact domains. Therefore, σ can be restricted to such domains so that σ is in L^p. This completes the proof. □

3.2.5 Learning dynamics by using the neural symplectic form

The proposed model learns the symplectic form and the Hamiltonian by minimizing the squared error between the left- and right-hand sides of (3.6), assuming that time series data of state vectors $\{u^{(n)}\}$ and its time-derivatives $\{\frac{\mathrm{d}u}{\mathrm{d}t}^{(n)}\}$ are given;

$$\text{minimize} \quad \sum_n \left| \frac{\mathrm{d}u}{\mathrm{d}t}^{(n)} - \tilde{W}^{-\top}_{u^{(n)}} \nabla H_{\mathrm{NN}}(u^{(n)}) \right|^2, \ (\tilde{W}_{u^{(n)}})_{i,j} = \frac{\partial (Y_{\mathrm{NN}})_i}{\partial u_j}(u^{(n)}) - \frac{\partial (Y_{\mathrm{NN}})_j}{\partial u_i}(u^{(n)}).$$

If the time derivatives are not available, interpolated data should be used. For each state vector $u^{(n)}$, first $Y_{\mathrm{NN}}(u^{(n)})$ is computed. Then, the derivatives of $Y_{\mathrm{NN}}(u^{(n)})$ are calculated by the automatic differentiation to obtain $(\tilde{W}_u^{-\top})_{i,j}$. Meanwhile, the gradient of the Hamiltonian is computed in the same way; first $H_{\mathrm{NN}}(^{(n)})$ is computed, and then $\nabla H_{\mathrm{NN}}(^{(n)})$ is obtained by the automatic differentiation.

In the standard Hamiltonian neural networks only the computation of the gradient of the Hamiltonian is required. Hence the computational cost of the neural symplectic form is roughly $2N$ times the computational cost of Hamiltonian neural networks, where $2N$ is the number of the state variables, because the number of components of $Y_{\mathrm{NN}}(u^{(n)})$ is $2N$. When N is large, such as in many-body problems, the most computationally expensive part may be computation of the inverse of the matrix $\tilde{W}_{u^{(n)}}$, which requires $O(N^3)$ operations. To avoid the computation of the inverse, one may be tempted to minimize

$$\sum_n \left| \tilde{W}_{u^{(n)}}^{\top} \frac{\mathrm{d}u^{(n)}}{\mathrm{d}t} - \nabla H_{\mathrm{NN}}(u^{(n)}) \right|^2 .$$

However, this model does not work because this is trivially minimized by setting $\tilde{W}_{u^{(n)}}^{\top} = O$ and $H_{\mathrm{NN}} =$ (constant.) Learning the inverse while maintaining the symplectic structure is future work.

3.3 Numerical Examples

The proposed method with the neural symplectic form can model general Hamiltonian equations on general symplectic manifolds using data represented by general coordinate systems. We illustrate these advantages in the following numerical experiments with comparative methods.

We performed the experiments using Hamiltonian neural networks, Lagrangian neural networks[*1], skew matrix learning and neural symplectic form. The energy function, the skew-symmetric matrix, and the 1-form for the neural symplectic 2-form were modeled by using a neural network that had two hidden layers of 200 units and the tanh activation function. We used 80 percent of collected data for training and the remaining for test. The collected data were normalized so that most of them were in the range $[-1, 1]$; however, the errors in this section are given in the original physical scale. It is worth noting that the collected data used in this experiment was obtained numerically and therefore there is some error in the simulated data compared

[*1] We used the implementation of the Lagrangian neural networks model in torchdyn [88] `https://github.com/DiffEqML/torchdyn/blob/master/docs/tutorials/09_lagrangian_nets.ipynb` (Apache 2.0 License)

to the actual real data. The neural symplectic form would be a Hamiltonian system, although the extent to which it deviates from the truth is related to the simulation errors in the data. We use the most commonly used numerical library SciPy, and the corresponding module for solving differential equations is scipy.integrate.solve_ivp, which uses the Runge-Kutta-Fehlberg method (denoted RKF45) in its default setting. Under this setting, the rerative error was set to 10^{-3} and hence the error for this experiment should be of the same order. We should use more precise data (eg. errors of 1e-6) for our experiments, which will be a topic of our future research.

We trained each model 10 times using the Adam optimizer with a learning rate of 10^{-3} for 2000 iterations. Only for Lagrangian neural networks, we set the learning rate to 10^{-4} due to an instability of learning; in our experiments the loss function of Lagrangian neural networks sometimes did not converge monotonically, but oscillated when the learning rate was set to 10^{-3}. Hence we truncated the training process at the iteration where the loss function achieved the best score. This oscillation should be due to the non-uniqueness of Lagrangian. See below for details.

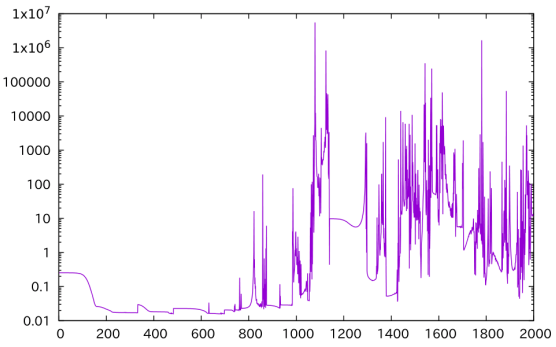


Fig.3.3 An example of the histories of the training loss of Lagrangian neural networks for the double pendulum data when the learning rate is set to 10^{-3}.

On the performances of Lagrangian neural networks In our experiments, although Lagrangian neural networks should be capable of modeling the targets, Lagrangian neural networks sometimes did not work well. Fig. 3.3 shows an example of the histories of the training loss of an Lagrangian neural network for the double pendulum data when the learning rate is set to 10^{-3}. As shown in the figure, the behavior of the loss function was not stable.

This may be due to the lack of enough data; in fact we use a much smaller data

than [20]. In addition, the neural network in [20] had three hidden layers with 500 units, whereas our neural network had two hidden layer with 200 units. The variance of the randomly initialized weight parameters was adjusted for each layer using more than 200 preliminary trainings. The dataset was composed of 600,000 orbits, whereas we used only 2,000 orbits. These differences should make the learning of Lagrangian neural network more stable than our experiments.

Another possibility is the non-uniqueness of the Lagrangian. In Lagrangian neural networks, the Lagrangian of the target dynamics is learned from the given data. Suppose that a Lagrangian $\mathcal{L}_{\mathrm{NN}}$, which is learned from the data, fits the given data. Then, the dynamics must be described by the Euler–Lagrange equation

$$\frac{\partial \mathcal{L}_{\mathrm{NN}}}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}_{\mathrm{NN}}}{\partial \dot{q}} = 0.$$

However, there are many Lagrangians that give the same Euler–Lagrange equation. In fact, the Euler–Lagrange equation is derived by the variational principle, in which the stationary value of the action integral

$$S = \int_0^T \mathcal{L}(q, \dot{q})\mathrm{d}t$$

with both ends fixed; however, even if the learned Lagrangian gives a stationary point, other Lagrangians can also give a stationary point. For example, for any smooth function $f(q)$, another Lagrangian $\tilde{\mathcal{L}} := \mathcal{L} + \mathrm{d}f/\mathrm{d}t$ gives the same Euler–Lagrange equation because the action integral associated with $\tilde{\mathcal{L}}$

$$\tilde{S} = \int_0^T \left(\mathcal{L}(q, \dot{q}) + \frac{\mathrm{d}f}{\mathrm{d}t}\right)\mathrm{d}t = \int_0^T \mathcal{L}(q, \dot{q})\mathrm{d}t + f(q(T)) - f(q(0)) = S + f(q(T)) - f(q(0))$$

takes the same stationary value as the original action integral S when both ends fixed. For example, two Lagrangians $\mathcal{L}$ and $\tilde{\mathcal{L}} = \mathcal{L} + q\dot{q}$ gives the same Euler–Lagrange equation because $q\dot{q} = \frac{1}{2}\frac{\mathrm{d}(q^2)}{\mathrm{d}t}$; the Euler–Lagrange equation of $\tilde{\mathcal{L}}$ is

$$\frac{\partial \tilde{\mathcal{L}}}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \tilde{\mathcal{L}}}{\partial \dot{q}}\right) = \frac{\partial \mathcal{L}}{\partial q} + \dot{q} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}} + q\right) = \frac{\partial \mathcal{L}}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}},$$

which is the Euler–Lagrange equation of $\mathcal{L}$. In Lagrangian neural networks, the Lagrangian is directly modeled by neural networks; however, because the Lagrangian is

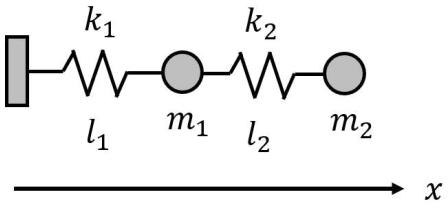


Fig.3.4 The mass-spring system.

not uniquely determined, even after a good Lagrangian is found, the learning algorithm goes on to find another Lagrangian which is possibly better. This is considered the reason for the instability of the learning processes of Lagrangian neural networks.

As a matter of fact, the loss function of Lagrangian neural network was unstable, but can be small. Therefore, in the numerical experiments, the model that achieved the minimum value of the histories was used to analyze the behavior of the models. In addition, the learning rate was set to a smaller value so that the training process could be more stable.

In the experiments, we implemented all code using Python v3.8.5 with libraries; numpy v1.21.2, scipy v1.7.1, and PyTorch v1.9.1. We performed all experiments on NVIDIA A100. After training, we evaluated the models using the squared time-derivative errors of the test subset. Using SciPy odeint under the default setting, we predicted 10 orbits from random initial values and obtained the errors in the system energy.

Mass-Spring System First, we investigate a Hamiltonian system in a general coordinate on a cotangent bundle, namely, a simple mass-spring system, depicted in Fig. 3.4. The equation of motion of this system is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} q_1 \\ q_2 \\ v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ -\frac{k_1}{m_1}(q_1 - l_1) + \frac{k_2}{m_1}(q_2 - q_1 - l_2) \\ -\frac{k_2}{m_2}(q_2 - q_1 - l_2) \end{pmatrix}, \tag{3.8}$$

which is a Hamiltonian system with the energy function

$$H(q_1, q_2, p_1, p_2) = \frac{p_1^2}{2m_1} + \frac{p_2^2}{2m_2} + \frac{k_1(q_1 - l_1)^2}{2} + \frac{k_2(q_2 - q_1 - l_2)^2}{2},$$

where q_1, q_2 are the positions of the mass points, and p_1, p_2 are the momenta, which are defined by $p_1 = m_1 v_1, p_2 = m_2 v_2, v_1 = \mathrm{d}q_1/\mathrm{d}t, v_2 = \mathrm{d}q_2/\mathrm{d}t$. Suppose that the

exact values of m_1 and m_2 are unknown, and only the positions q_1 and q_2 and their derivatives are given. Although m_1 and m_2 may be estimated from the given states, for evaluation of the models, we tried to learn the dynamics from the given states directly.

Note that it should be impossible to write the equation (3.8) as the standard form of Hamiltonian equation (3.2) with a certain energy function; for example, when the system has just one mass point and the equation of motion is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} q_1 \\ v_1 \end{pmatrix} = \begin{pmatrix} v_1 \\ -\frac{k_1}{m_1}(q_1 - l_2) \end{pmatrix},$$

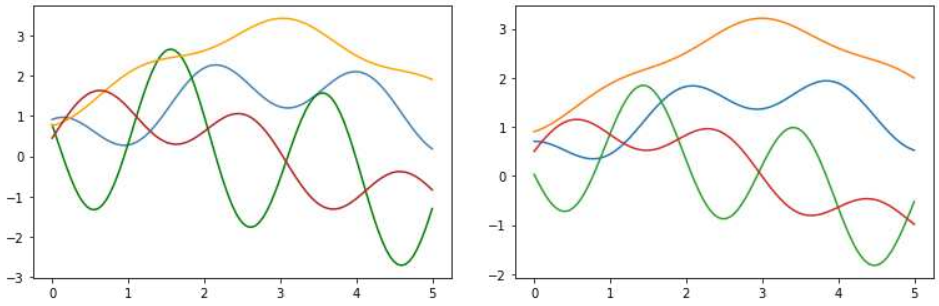
this can be transformed into a Hamiltonian system

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} q_1 \\ v_1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \nabla \tilde{H},$$
$$\tilde{H} = \frac{v_1^2}{2m_1'} + k_1'(q_1 - l_1)^2$$

with $k_1' = k_1/m_1, m_1'$ and the mass m_1' is 1. Hence, for this system, Hamiltonian neural networks are applicable without knowledge of m_1. However, for the above system (3.8), such a transformation cannot be applied.

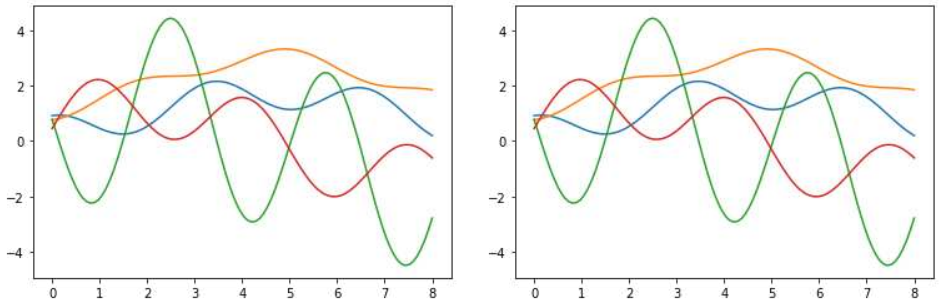
The detailed setup of the experiment is as follows. As the data, we used numerical solutions to (3.8)with the parameters $k_1 = 3.0$, $k_2 = 5.0$, $l_1 = 1.0$, $l_2 = 1.0$, $m_1 = 1.0$, $m_2 = 2.0$. The initial values are randomly sampled from the standard normal distribution. The numerical solutions are computed on the time interval $[0, 5]$. For each numerical orbit, 100 solutions are sampled at uniform time intervals. The numerical solutions are by using SciPy odeint with the default setting.

The time-derivative errors and the energy errors are shown in Tables 3.2 and 3.3, respectively. Nevertheless, as this is a simple problem, all the models work well. Most of the predicted orbits shown in Fig. 3.5 have a good fit to the true one. Note that the errors shown in Table 3.2 are small enough. They are multiplied by 10^3 and shown in the physical scale; i.e., the errors are not measured by using the normalized state variables but by using the state variables with the true scale. In the normalized scale, the error by Lagrangian neural network, for example, was 0.0095. This is also the case in the other experiments. Among the methods, neural ordinary differential equations and the neural symplectic form performed better. The test time-derivative



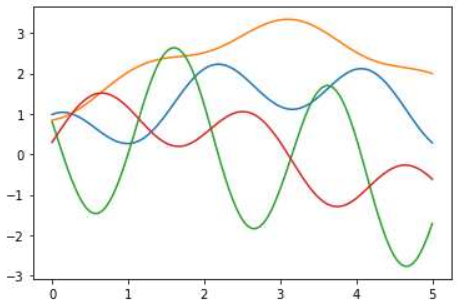
(a) Ground truth

(b) NODE

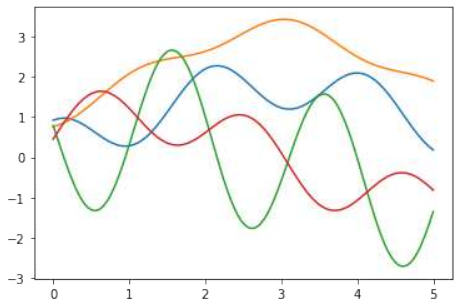


(c) HNN

(d) LNN



(e) Skew Matrix Learning

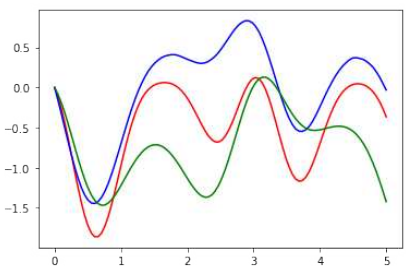


(f) Neural Symplectic Form

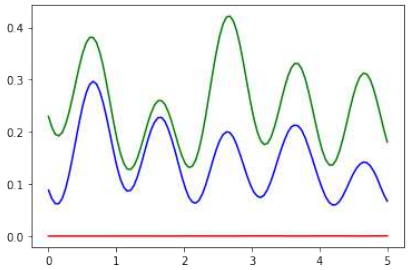
Fig.3.5 Example of the orbits predicted by the trained models for the mass-spring test. The horizontal axis represents time. Each component of $u(t) = (q_1(t), v_1(t), q_2(t), v_2(t))$ is represented: blue (q_1), green (v_1), orange (q_2), and red (v_2).

error of the skew matrix learning is a little larger. In terms of the energy error, the neural symplectic form and Hamiltonian neural networks are superior; however there is not much difference.

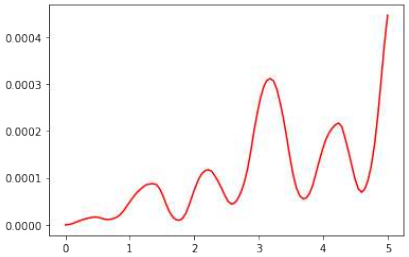
Fig. 3.6 shows the time evolution of the errors for neural ordinary differential equations, the skew matrix learning and the neural symplectic form. The corresponding



(a) Energy errors



(b) Solution errors



(c) Solution error of Neural Symplectic Form

Fig.3.6 Time evolution of the energy and solution errors obtained by NODE, the skew matrix learning and the neural symplectic form for the mass-spring test. The horizontal axis represents time. The energy error shows the difference from the true energy, and the solution error shows the MSEs. Since the solution error by the neural symplectic form was tiny, the enlarged graph is also shown.

orbits are shown in Fig. 3.5. In this case, the prediction of the neural symplectic form was very accurate. This may be related to the fact that the proposed method is able to preserve the conservation laws other than energy. In fact, in the research field of physical simulation, it is known that for such a small system, the computational accuracy can be often improved if additional conservation laws are preserved. As for the energy error, there is not much difference between the methods. The energy error appears to be large compared to the solution error; this should be due to the numerical integration errors in the computation of the predicted orbits.

Double Pendulum Next, we consider another Hamiltonian system, namely, a double pendulum shown in Fig. 3.7, of which equation of motion is

$$\begin{aligned}
&\frac{\mathrm{d}\theta_1}{\mathrm{d}t} = \phi_1, \quad \frac{\mathrm{d}\theta_2}{\mathrm{d}t} = \phi_2, \\
&\frac{\mathrm{d}\phi_1}{\mathrm{d}t} = \frac{g(\sin\theta_2 \sin(\theta_1-\theta_2) - \frac{m_1+m_2}{m_2}\sin\theta_1) - (l_1\theta_1^2\cos(\theta_1-\theta_2) + l_2\theta_2^2)\sin(\theta_1-\theta_2)}{l_1(\frac{m_1+m_2}{m_2} - \cos^2(\theta_1-\theta_2))}, \\
&\frac{\mathrm{d}\phi_2}{\mathrm{d}t} = \frac{\frac{g(m_1+m_2)}{m_2}(\sin\theta_1\cos(\theta_1-\theta_2) - \sin\theta_2) - (\frac{l_1(m_1+m_2)}{m_2}\theta_1^2 + l_2\theta_2^2\cos(\theta_1-\theta_2))\sin(\theta_1-\theta_2)}{l_2(\frac{m_1+m_2}{m_2} - \cos^2(\theta_1-\theta_2))}.
\end{aligned} \tag{3.9}$$

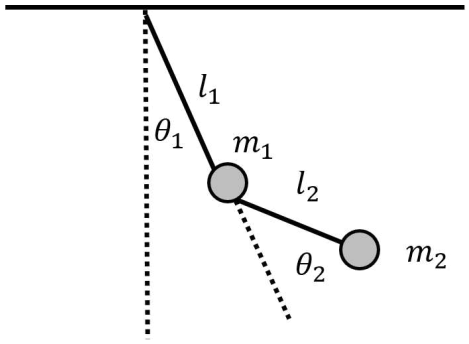


Fig.3.7 A double pendulum and the related constants and the variables.

The energy function of this system is

$$\begin{aligned}
H =& \frac{1}{2}(m_1+m_2)l_1^2\phi_1^2 + \frac{1}{2}m_2l_2^2\phi_2^2 + m_2l_1l_2\phi_1\phi_2\cos(\theta_1-\theta_2) + gm_2l_2\cos(\theta_2) \\
&+ g(m_1+m_2)l_1\cos\theta_1
\end{aligned}$$

and the Lagrangian is

$$\begin{aligned}
\mathcal{L} =& \frac{1}{2}(m_1+m_2)l_1^2\phi_1^2 + \frac{1}{2}m_2l_2^2\phi_2^2 + m_2l_1l_2\phi_1\phi_2\cos(\theta_1-\theta_2) - gm_2l_2\cos(\theta_2) \\
&- g(m_1+m_2)l_1\cos\theta_1,
\end{aligned}$$

which derives the generalized momentum

$$\begin{aligned}
p_1 &= \frac{\partial\mathcal{L}}{\partial\phi_1} = (m_1+m_2)l_1^2\phi_1 + m_2l_1l_2\phi_2\cos(\theta_1-\theta_2), \\
p_2 &= \frac{\partial\mathcal{L}}{\partial\phi_2} = m_2l_2^2\phi_2 + m_2l_1l_2\phi_1\cos(\theta_1-\theta_2).
\end{aligned}$$

Because the generalized momenta are not obvious, we assume that the data of $\theta_1, \theta_2, \phi_1, \phi_2$ are given instead of $\theta_1, \theta_2, p_1, p_2$, where ϕ_1, ϕ_2 are the time derivatives of θ_1, θ_2.

Table3.2 Test time-derivative errors.

	NODE	HNN	LNN	Skew Matrix Learning	Neural Symplectic Form (proposed)
mass-spring	**0.17 ± 0.14**	694.77 ± 26.37	882.52 ± 1753.93	102.07 ± 68.00	<u>0.52 ± 0.71</u>
double pendulum	8.65 ± 1.38	15.69 ± 0.44	269.52 ± 136.85	<u>8.47 ± 1.22</u>	**4.02 ± 2.08**
Lotka–Volterra	2.02 ± 5.55	227.03 ± 2.31	N/A	<u>1.05 ± 0.76</u>	**0.46 ± 0.30**

The best and second best results are emphasized by bold and underlined fonts, respectively. Multiplied by 10^3 for the mass-spring system and the Lotka–Volterra equation. The experiments were conducted 10 times each. The results of Lagrangian neural network for double pendulum were computed using nine times except for a failed one.

We collected data by solving (3.9) with the parameters $l_1 = l_2 = 1.0$, $m_1 = 1$, $m_2 = 2$, and $g = 9.8$ using 2000 initial conditions randomly generated from the standard normal distribution. The batch-size was set to 1000.

The test time-derivative errors are shown in Table 3.2. Firstly, (3.9) cannot be written as the standard form (3.2) with a certain Hamiltonian. Hence, the test time-derivative error of Hamiltonian neural networks cannot be completely zero when learned using states $\theta_1(t), \phi_1(t), \theta_2(t), \phi_2(t)$. The result shown in Table 3.2 confirms this. The test time-derivative errors by neural ordinary differential equations, the skew matrix learning and the neural symplectic form are much smaller than that by Hamiltonian neural networks. In particular, the error by the skew matrix learning is larger than that by the neural symplectic form. This is due to the non-existence of the one-to-one correspondence between the skew matrix and the symplectic 2-form, which decreases the efficiency of learning. neural ordinary differential equations performed very well too; however, as explained below, this model failed to predict long-term behaviors.

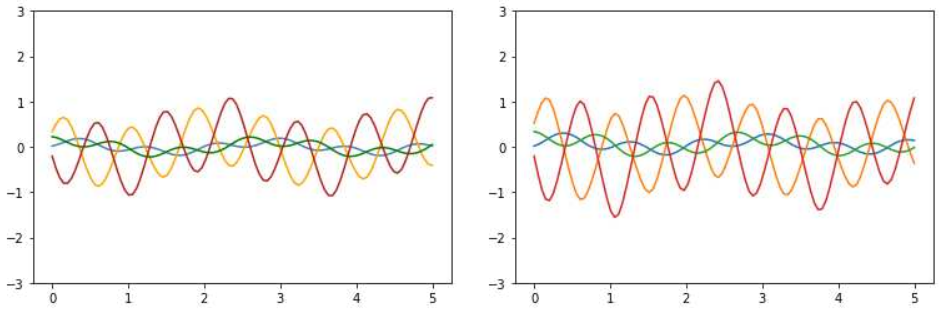
Table3.3 Energy errors.

	NODE	HNN	LNN	Skew Matrix Learning	Neural Symplectic Form (proposed)
mass-spring	0.840 ± 0.328	0.551 ± 0.112	2.281 ± 0.004	6.203 ± 7.555	**0.368 ± 0.055**
double pendulum (T=5)	**1.070 ± 0.694**	(0.755 ± 0.320)	17.740 ± 10.804	3.931 ± 7.266	6.400 ± 0.971
double pendulum (T=30)	11.240 ± 12.297	N/A	N/A	622982.133 ± 1814794.079	**7.300 ± 3.925**
Lotka–Volterra	0.578 ± 0.558	0.444 ± 0.458	N/A	0.041 ± 0.072	**0.012 ± 0.013**

The best and second best results of the true energies are emphasized by bold and underlined fonts, respectively. The differences of the true energy functions at $t = T$ and $t = 0$, where $T = 30$ for the Lotka–Volterra equation and $T = 5$ for the others. 10 orbits were simulated using randomly generated initial values. In the double pendulum test, the energy error of the Hamiltonian neural network was best but this is because of the very small amplitudes of the predicted solutions; hence the result of NODE is cosidered best.

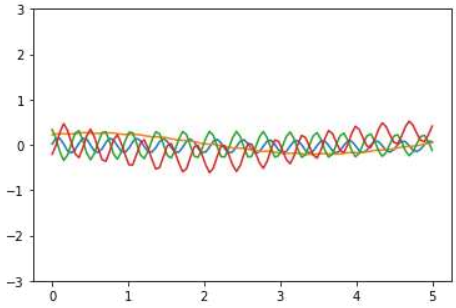
Examples of the predicted orbits are shown in Fig. 3.8. As expected, Hamiltonian neural networks failed to predict the orbits. In the result by the skew matrix learning, although the speed of oscillation appears to be correct, the heights of the peaks are different from the true trajectory. Meanwhile, the prediction by the neural symplectic form achieved a better agreement with the true one than other models.

The energy errors are shown in Table 3.3. The errors of neural ordinary differential equations and the skew matrix learning are small for the short term prediction. However, Fig. 3.9 shows an example of the long-term prediction results of these models and the proposed neural symplectic form. The results of the two neural ordinary differential equations are obtained using the models trained with different seeds. Since neural ordinary differential equations does not have a Hamiltonian structure, the results gradually decay or diverge. The prediction results by the skew matrix learning also show a gradual increase in the state variables, and the results collapse after exceeding a certain value. On the other hand, the result by the neural symplectic form

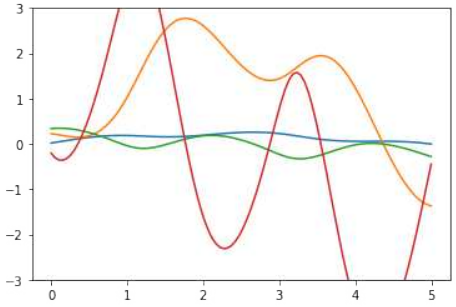


(a) Ground truth

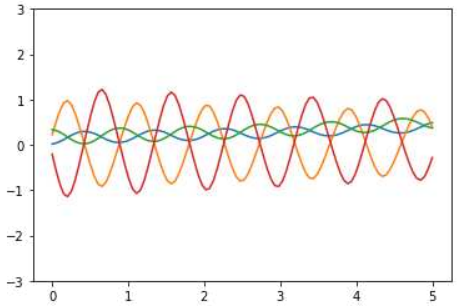
(b) NODE



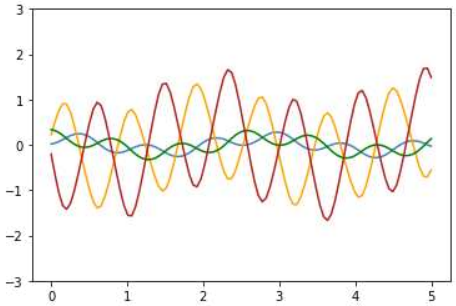
(c) HNN



(d) LNN



(e) Skew Matrix Learning

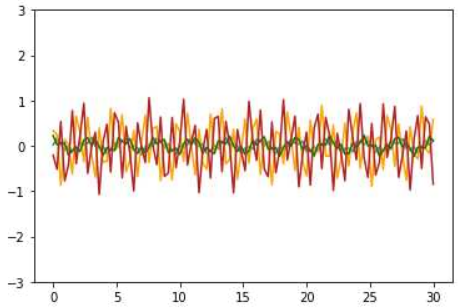


(f) Neural Symplectic Form

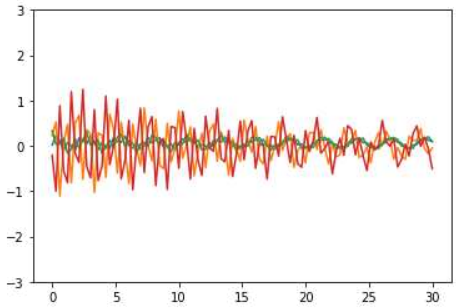
Fig.3.8 Example of the orbits predicted by the trained models for the double pendulum test. The horizontal axis represents time. Each component of $u(t) = (\theta_1(t), \phi_1(t), \theta_2(t), \phi_2(t))$ is represented: blue (θ_1), orange (ϕ_1), green (θ_2), and red (ϕ_2).

oscillates stably, although the range of oscillation is a little larger than the true orbit.

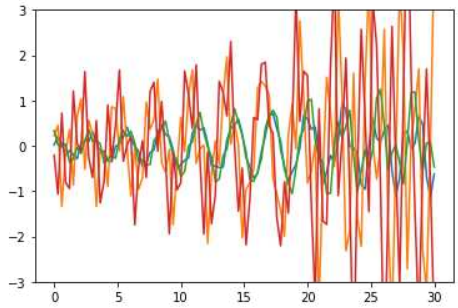
We show the time evolution of the errors for the neural ordinary differential equations, the skew matrix learning and the neural symplectic form in Fig. 3.10 along with the corresponding orbits in Fig. 3.8. The errors are not significantly different;



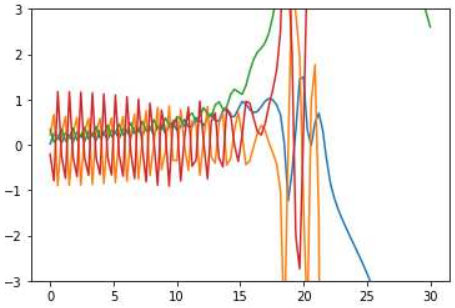
(a) Ground truth



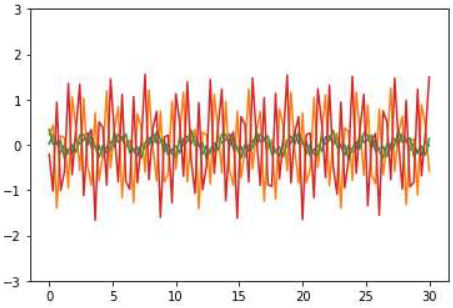
(b) NODE1



(c) NODE2



(d) Skew Matrix Learning

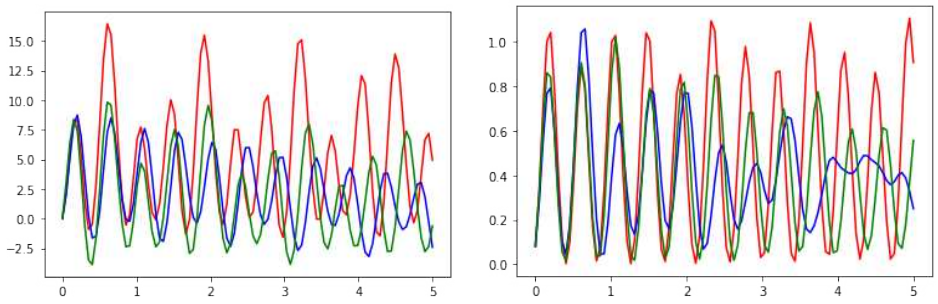


(e) Neural Symplectic Form

Fig.3.9 Example of the orbits during a long period ($t = 30$) predicted by the trained models for the double pendulum test. NODE1 and NODE2 are results of two NODE models trained with a different random seed. The horizontal axis represents time. Each component of $u(t) = (\theta_1(t), \phi_1(t), \theta_2(t), \phi_2(t))$ is represented: blue (θ_1), orange (ϕ_1), green (θ_2), and red (ϕ_2).

however, the errors for the neural ordinary differential equations and the skew matrix learning are relatively small. Although these methods are not suitable for long-term predictions, they are effective for short-term predictions.

We also performed a test for a more chaotic orbit using the neural ordinary differ-



(a) Energy errors (b) Solution errors

Fig.3.10 Time evolution of the energy and solution errors obtained by NODE, the skew matrix learning and the neural symplectic form for the double-pendulum test. The errors are represented: blue (skew matrix learning), green (NODE), red (neural symplectic form). The horizontal axis represents time. The energy error shows the difference from the true energy, and the solution error shows the MSEs.

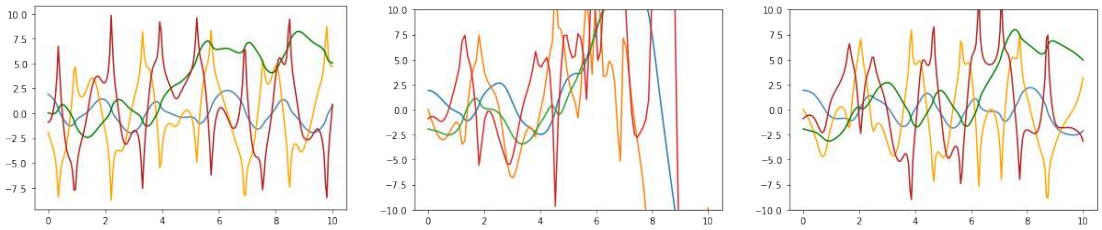
ential equations and the neural symplectic form. Examples of the predicted orbits are shown in Fig. 3.11. Note that in this experiment, we used models trained with the same data as the above experiment, and this data may not contain many chaotic trajectories.

Because the results of the neural ordinary differential equations often diverged as shown in this figure we omit the quantitative results for this test. Although the predicted orbits are not so similar to the true one, the proposed method typically kept oscillating. The worse performance may be due to the lack of enough data on the behaviors when the angle became large. In fact, a chaotic double pendulum can rotate many times, and its angle can be very large; however, our data are generated by simulating the orbits from relatively small initial conditions. For better performance, the periodic structure of the phase space as a Lie group should be integrated into the model for efficient training.

Lotka–Volterra equation Systems of differential equations with a polynomial right-hand side are often Hamiltonian with a hidden symplectic structure. For example, the Hamiltonian structure of the generalized Lotka–Volterra equation

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = x_i\Big(\sum_{j=1}^{m} a_{ij}\Pi_{k=1}^{l} x_k^{b_{jk}} + \lambda_i\Big), \tag{3.10}$$

is investigated in [42]. The right-hand side of this equation is quite general because this



(a) Ground truth (b) NODE (c) Neural Symplectic Form

Fig.3.11 Example of the orbits predicted by the trained models for the double pendulum with a chaotic behavior. The horizontal axis represents time. Each component of $u(t) = (q_1(t), v_1(t), q_2(t), v_2(t))$ is represented: blue (q_1), green (v_1), orange (q_2), and red (v_2).

equation has the form of x_i(polynomial of the other state variables). For example, many mathematical compartment models in biology are of this form. In fact, the original Lotka–Volterra model is proposed as a model of a predator-prey dynamics.

We used a standard Lotka–Volterra model, which is included in (3.10), of the following form:

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = a_{12}x_1x_2 + \lambda_1 x_1, \quad \frac{\mathrm{d}x_2}{\mathrm{d}t} = a_{21}x_1x_2 + \lambda_2 x_2.$$

Provided that $x_1 \neq 0$ and $x_2 \neq 0$, this equation can be written as a Hamiltonian equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} O & x_1x_2 \\ -x_1x_2 & O \end{pmatrix}\begin{pmatrix} \frac{\partial H}{\partial x_1} \\ \frac{\partial H}{\partial x_2} \end{pmatrix}, \quad H(x_1, x_2) = -a_{21}x_1 - \lambda_2 \ln x_1 + a_{12}x_2 + \lambda_1 \ln x_2,$$

which is different from the standard Hamiltonian equation (3.2). In fact, this equation cannot be written as (3.2) globally; the Darboux coordinate system only *locally* exists in general. For example, if the first equation can be written as

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = \frac{\partial H}{\partial x_2}$$

with a certain function $H(x_1, x_2)$, $H(x_1, x_2)$ should be of the form

$$H(x_1, x_2) = \frac{a_{12}}{2}x_1x_2^2 + \lambda_1 x_1x_2 + f(x_1)$$

with a function f. However, in that case it must hold that

$$\frac{\partial H(x_1, x_2)}{\partial x_1} = \frac{a_{12}}{2}x_2^2 + \lambda_1 x_2 + \frac{\mathrm{d}f(x_1)}{\mathrm{d}x_1},$$

which cannot be in general the right-hand side of the second equation of the Lotka–Volterra equation for any f. In the experiment, we checked if this unknown symplectic structure can be extracted from the data or not.

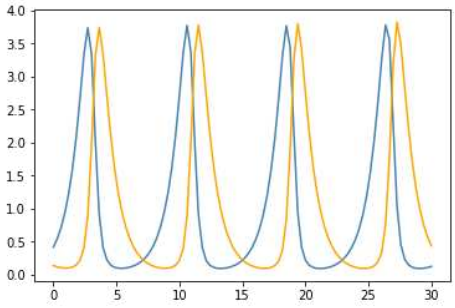
In the experiment, because the state variables are not the pair of q and $\dot{q}$, Lagrangian neural networks is not applicable to this equation. Hence, we tested neural ordinary differential equations, Hamiltonian neural networks, the skew matrix learning and the neural symplectic form. We set the parameters as $a_{12} = -1, a_{21} = -1, \lambda_1 = 1$ and $\lambda_2 = 1$. As the data, 1000 orbits on the time interval $[0, 5]$ are numerical computed by using SciPy odeint. Initial conditions are sampled from the uniform distribution on $[0, 1]$. Each orbit contains 100 numerical solutions at a uniform sampling rate. The data are normalized so that they are in $[0, 1]$.

The test time-derivative errors are shown in Table 3.2. Again, the error by Hamiltonian neural networks is larger than those by the other models. As seen from the table, the proposed neural symplectic form stably gave better results than the other models.

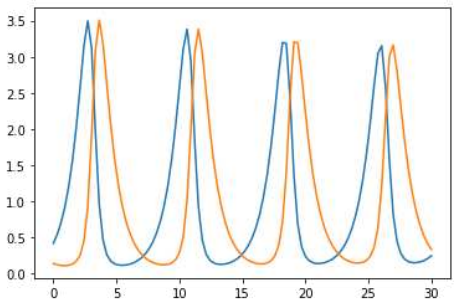
Regarding the energy errors, the energies are well preserved as shown in Table 3.3. In particular, the proposed method preserves the energy with the highest accuracy. Besides, predicted orbits are shown in Fig. 3.12. The peaks of the orbits by Hamiltonian neural networks and the skew matrix learning are smaller than the true trajectory, and the orbit of the neural ordinary differential equations is gradually decaying. On the other hand, the proposed neural symplectic form gives the almost identical orbits to the true one.Thus the hidden symplectic structure of this equation is certainly extracted by the proposed method.

We show the time evolution of the errors for the neural ordinary differential equations, the skew matrix learning and the neural symplectic form in Fig. 3.13. The corresponding orbits are shown in Fig. 3.12. Since the peaks of the orbit of the neural ordinary differential equations is gradually decaying, the energy error of the neural ordinary differential equations is increasing. Regarding the solution errors, although the predicted orbit of the skew matrix learning is not so different from the ground truth, the error becomes very large. This is due to the error in the velocity of the oscillation. This error causes the position of the peaks to deviate from its true position, resulting in a large error.

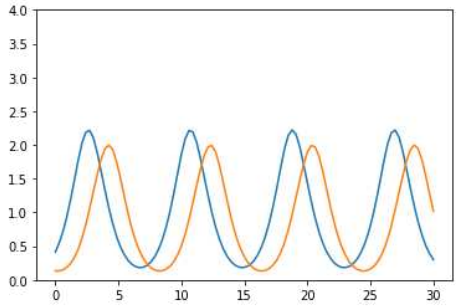
In addition, we also performed another test in which long term behaviors are pre-



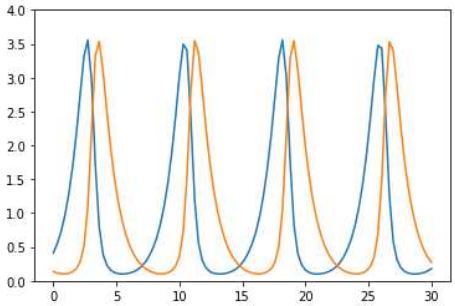
(a) Ground truth



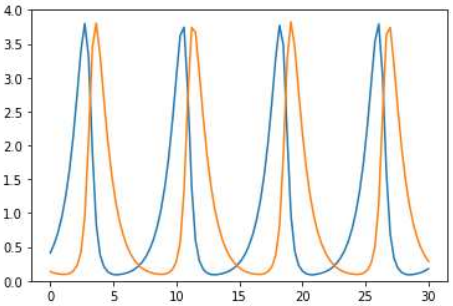
(b) NODE



(c) HNN



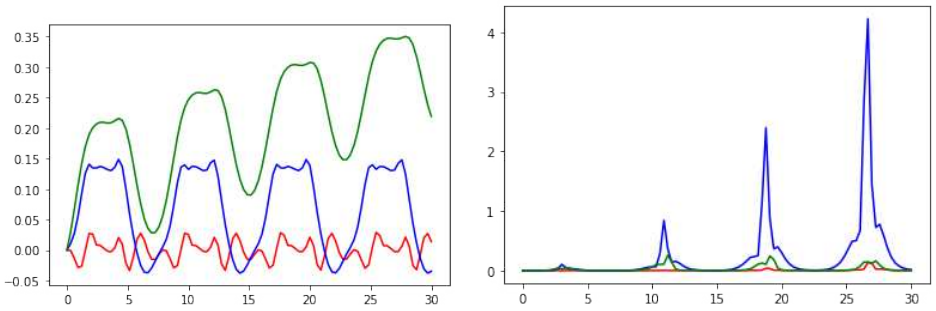
(d) Skew Matrix Learning



(e) Neural Symplectic Form

Fig.3.12 Example of the orbits predicted by the proposed and the comparative models for the Lotka–Volterra model. The horizontal axis represents time. $x_1(t), x_2(t)$ are represented: blue (x_1) and orange (x_2).

dicted. In this case, we compared the well-performed models: neural ordinary differential equations, skew matrix learning and the neural symplectic form. The predicted orbits are shown in Fig. 3.14. The two neural ordinary differential equations results are obtained by the two neural ordinary differential equations trained with a different random seed. Because the neural ordinary differential equations does not satisfy



(a) Energy errors (b) Solution errors

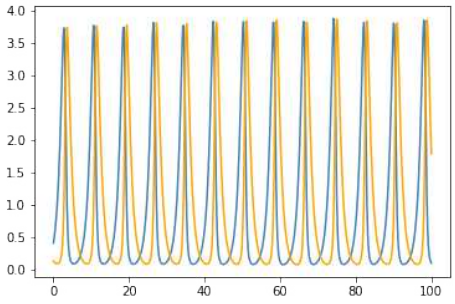
Fig.3.13 Time evolution of the energy and solution errors obtained by NODE, the skew matrix learning and the neural symplectic form for the Lotka–Volterra test. The errors are represented: blue (skew matrix learning), green (NODE), red (neural symplectic form). The horizontal axis represents time. The energy error shows the difference from the true energy, and the solution error shows the MSEs.

the energy conservation law, the predicted orbits by these models gradually decay or diverge as expected. Meanwhile, the orbits by the other two models keep oscillating. The energy errors are shown in Table 3.4. While the error of the neural ordinary differential equations is very large, the results by the other two are almost the same. The error of the skew matrix learning is slightly larger; this is probably due to the lower height of the peaks.

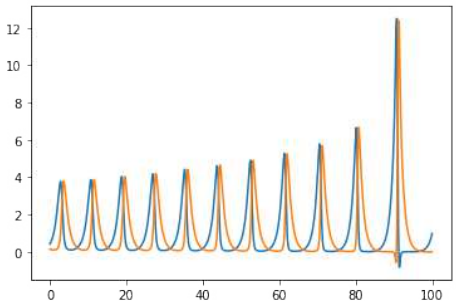
Table3.4 Energy errors.

	NODE	Skew Matrix Learning	Neural Symplectic Form (proposed)
Lotka–Volterra ($t = 100$)	42.951 ± 56.005	<u>0.0735 ± 0.0764</u>	**0.0704 ± 0.0597**

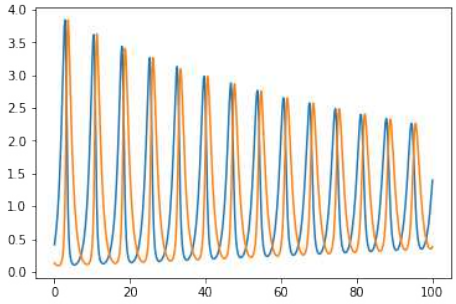
The best and second best results of the true energies are emphasized by bold and underlined fonts, respectively. The differences of the true energy functions at $t = 100$ and $t = 0$ are shown. 10 orbits were simulated using randomly generated initial values.



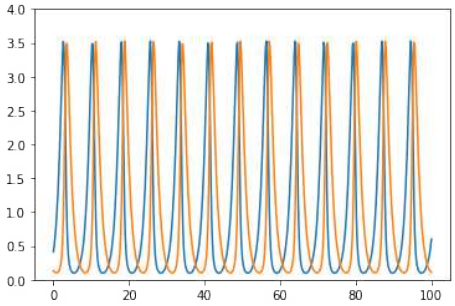
(a) Ground truth



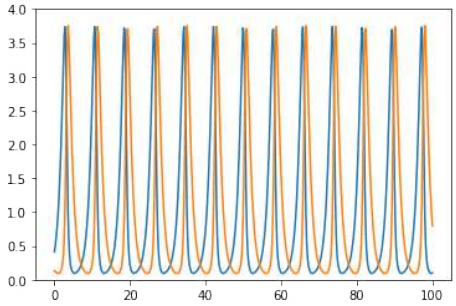
(b) NODE1



(c) NODE2



(d) Skew Matrix Learning



(e) Neural Symplectic Form

Fig.3.14 Example of the orbits during a long period ($t = 100$) predicted by the trained models for the Lotka–Volterra test. NODE1 and NODE2 are results of two NODE models trained with a different random seed. The horizontal axis represents time. $x_1(t)$, $x_2(t)$ are represented: blue (x_1), orange (x_2).

Learning from Images As an application of the proposed method, we learned the equation of motion from some images. One way to achieve this is to extract features from the images by using an autoencoder and to learn the equation of motion that the features satisfy [38]. In this case, the extracted features are not supposed to be momenta. The proposed method is suitable for this application because it is

independent of the coordinate system.

In this experiment, we first pretrained an autoencoder. Then Hamiltonian neural networks and the neural symplectic form are applied to learn the dynamics of the features. In this experiment, we used Hamiltonian neural networks code and data (`https://github.com/greydanus/hamiltonian-nn`, Apache 2.0 License) almost as-is; we used a pre-trained autoencoder with two hidden layers of 200 units and the ReLU activation function. We trained the autoencoder using the Adam optimizer with a learning rate of 10^{-4} for 150000 iterations. The test error of the autoencoder was 3.81e-03. The neural networks in Hamiltonian neural networks and the neural symplectic forms have two hidden layers of 200 units and the tanh activation function. These networks are trained for 100000 steps; in our experiment it took a long time for the neural symplectic form to find an appropriate symplectic form. The original Hamiltonian neural networks code includes a loss function that measures how close the latent space is to the canonical coordinates in order to make it closer to the canonical coordinates. However, in our experiment, we trained without this loss function. The other settings are the same as in the similar experiment of Hamiltonian neural networks. The final test losses of Hamiltonian neural networks and the neural symplectic form were 0.161 and 0.060 respectively. The predicted pictures are shown in Fig. 3.15. The last four pictures of Hamiltonian neural networks are noisy; this implies relatively large errors in the latent space. However, this is only a preliminary test; for example, the performance may depend on the architecture of the autoencoder. Further thorough investigation is needed for this application.

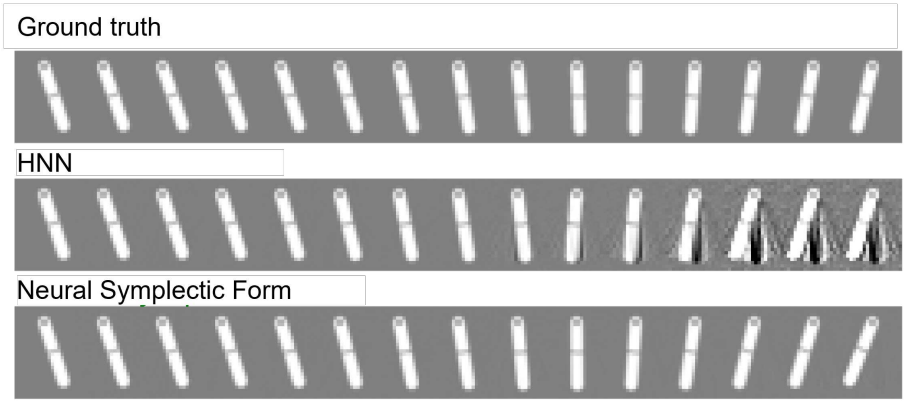


Fig.3.15 Example of the predicted images by the proposed method and Hamiltonian neural networks.

Chapter 4 KAM Theory Meets Statistical Learning Theory: Hamiltonian Neural Networks with Non-Zero Training Loss

In this chapter, we consider the equation of Hamiltonian mechanics in the form

$$\frac{\mathrm{d}u}{\mathrm{d}t} = S\frac{\partial H}{\partial u}, \tag{4.1}$$

where $u : t \in \mathbb{R} \mapsto u(t) \in \mathbb{R}^N$, $H : u \in \mathbb{R}^N \mapsto H(u) \in \mathbb{R}$ and S is a skew-symmetric matrix. The Hamiltonian neural network which is a typical deep physical model for learning the above equation was proposed:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = S\frac{\partial H_{\mathrm{NN}}}{\partial u}. \tag{4.2}$$

In this chapter, we focus on the Hamiltonian neural network, particularly in practical situations where the learning error is not completely zero. In this case, the trained model can be regarded as a perturbed Hamiltonian system due to the modelling error of the energy function. In addition, S is a general skew-symmetric matrix and hence (4.2) includes discretizations of Hamiltonian partial differential equations [75].

In mathematical physics, the recursive nature of physical phenomena is also of interest in physics; however, whether such phenomena can be reproduced by deep learning models is an important problem that greatly affects the usefulness of deep physical models. In this chapter, we give an answer to this question by combining the KAM theory and statistical learning theory.

Importantly, for the neural network models to be close to the true dynamics, we need a universal approximation theorem and also a generalization error bound. In this chapter, we also provide such results for Hamiltonian neural networks. Regarding

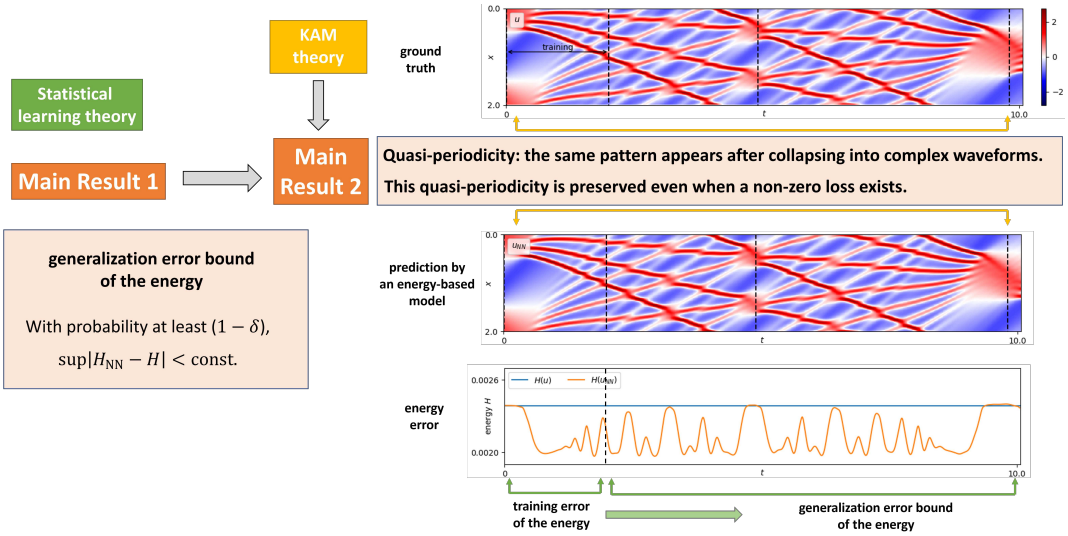


Fig.4.1 Outline of our main theorem. The first main result is the generalization error of the energy function, which is proved by the statistical learning theory. By combining the first result and the KAM theorem, we prove that the quasi-periodic behaviors of target systems are preserved.

the generalization error bound, because the derivative of a multi-layer perceptron is used in Hamiltonian neural networks, a bound for the derivative is required. To this end, we estimated the covering number of the derivative of multi-layer perceptrons. An L^∞ bound on the error in the Hamiltonian is also provided, which is required for application of the KAM theory.

In addition, we show a universal approximation theorem for a model with the coordinate transformation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = (\frac{\partial u}{\partial x})^{-1} S (\frac{\partial u}{\partial x})^{-\top} \frac{\partial H}{\partial x} \approx (\frac{\partial u_{\mathrm{NN}}}{\partial x})^{-1} S (\frac{\partial u_{\mathrm{NN}}}{\partial x})^{-\top} \frac{\partial H_{\mathrm{NN}}}{\partial x}. \tag{4.3}$$

This model was indispensable in practice; to apply Hamiltonian neural networks, the data are given in the canonical coordinate system because the equation of motion is in the form of Hamilton equation (4.1) only when the state variables are represented by the canonical coordinate system. However, this problem has been addressed in Chapter 3.

The main contributions of this study are as follows:

1. Combination of the KAM theory and statistical learning theory for Hamiltonian neural networks with non-zero training loss to prove the existence of quasi-periodic behaviors (see Fig. 4.1).

2. Derivation of a generalization error bound for Hamiltonian neural networks.

3. Development of a universal approximation theorem for Hamiltonian neural networks and other energy-based physical models with coordinate transformations.

This chapter is organized as follows. First, in Section 4.1, we first explain existing work on the theoretical study of Hamiltonian neural networks and their extension models, as well as the uniqueness of the theoretical study conducted in this chapter. In Section 4.2, we give a brief introduction to Hamiltonian systems and the KAM theory. In Section 4.3, we show the main results in which the universal approximation properties and the generalization performance of deep physical models are investigated. In addition, the application of this result to the KAM theory is also provided.

4.1 Preparation

Many physical phenomena are described by Hamiltonian mechanics using an energy function. Recently, the Hamiltonian neural network, which approximates the Hamiltonian by a neural network, and its extensions have attracted much attention. Before presenting the results of this theoretical study, we will give an introduction to the previous studies related to it.

Meanwhile, the model (4.3) is indispensable in practice; to apply Hamiltonian neural networks, the data are given in the canonical coordinate system because the equation of motion is in the form of the Hamilton equation (4.1) only when the state variables are represented by the canonical coordinate system. However, this coordinate system depends on an unknown Lagrangian and hence the energy function. Hence, the coordinate system must be also learned from data by using, for example, neural networks. In addition, this model also represents other energy-based physical models beyond the Hamilton equation. We will explain this in more detail in this chapter.

4.1.1 Related work

Many studies of neural network models for physical phenomena that can be modeled by energy-based equation (4.1) have been put forward. Among them, the most basic studies are neural ordinary differential equations [15] and Hamiltonian neural networks [38]. In particular, extensions of Hamiltonian neural networks have been intensively developed.

Describing them all is beyond the scope of thsi chapter, but some examples are given here. In [116] Hamiltonian neural networks were extended to latent variable models. Other studies, such as [25, 126, 135], focused on the symplectic structure of the Hamilton equation. For Noether's theorem, which is a fundamental theorem in classical mechanics, several studies [6, 7, 31] developed methods related to symmetry and conservation laws. In addition, a discrete-time model that preserves the energy behaviors was constructed in [75]. In [34], Hamiltonian neural networks were combined with a Bayesian approach.

Methods applied to the framework of classical mechanics other than Hamiltonian mechanics include those in [20, 24, 98], which are methods for Lagrangian formalism. In [52], reinforcement learning was applied to the variational principle. A simplified model formed by introducing constraints was proposed in [32]. In [50], Hamiltonian neural networks were extended to the Poisson system, which is a wider class of mechanical equations. There are also a number of proposals that integrate them with more advanced deep learning techniques, namely, graph networks [100], recurrent neural networks [19], and normalizing flows [66]. As an application-oriented approach, [30] designed a microscopic model for structural analysis.

However, to the author' s best knowledge, there is no theoretical research other than the universal approximation theorems for Hamiltonian mechanics in SympNet [51], in which a certain kind of neural network is shown to have universal approximation properties for symplectic maps. The difference between their results and ours is that (1) we analyze the behaviors of a Hamiltonian neural network with non-zero training loss by a combination of the KAM theory and statistical learning theory, (2) we provide a generalization error bound for Hamiltonian neural networks, and (3) the universal approximation theorems in [51] are for discrete-time models, while ours are for continuous-time models.

Meanwhile, as an existing energy-based model, the Hopfield neural network is known. Both the Hopfield neural network and Hamiltonian neural networks are derived from energy-based theories, and their dynamics are described by (1). The Hamiltonian neural network is associated with a skew-symmetric matrix S and is a model of an energy-preserving, continuous-time, and deterministic physics phenomenon. Its output is the time-series of the state. Hopfield network is associated with a negative definite matrix S and exhibits a dynamics which is often energy-dissipating, discrete-time, and stochastic. It is a machine learning tool rather than a physical model, and its equilibrium point is treated as its output. Because their outputs are different, their theoretical properties should be discussed separately.

4.1.2 Energy-based physical models

Equation (4.1) (and its coordinate transformation (4.3)) and the related model (4.2) (and its coordinate transformation (4.3)) describe not only Hamiltonian systems but also various types of other phenomena. In fact, as shown in the theorem below, models similar to (4.1),

$$\frac{\mathrm{d}u}{\mathrm{d}t} = G\frac{\partial H}{\partial u}, \tag{4.4}$$

where G is a skew-symmetric or negative-semidefinite matrix, have not only the energy conservation property but also the energy-dissipation property, depending on the matrix G. The same result for the models similar to (4.3) is obtained in a straightforward way.

Theorem 4.1. Equation (4.4) admits the energy conservation law,

$$\frac{\mathrm{d}H}{\mathrm{d}t} = 0,$$

if G is skew-symmetric, and it admits the energy dissipation law,

$$\frac{\mathrm{d}H}{\mathrm{d}t} \leq 0,$$

if G is negative-semidefinite.

Proof of Theorem 4.1 We use the chain rule to obtain

$$\frac{\mathrm{d}H}{\mathrm{d}t} = \frac{\partial H}{\partial u}^{\top}\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{\partial H}{\partial u}^{\top} G\frac{\partial H}{\partial u}. \tag{4.5}$$

Hence, when the matrix G is skew-symmetric, the energy conservation law holds:

$$\frac{\mathrm{d}H}{\mathrm{d}t} = 0$$

because $v^\top Gv = 0$ holds for all vectors v when G is skew-symmetric.

When the matrix G is negative semidefinite, the energy function is monotonically non-increasing:

$$\frac{\mathrm{d}H}{\mathrm{d}t} = \frac{\partial H}{\partial u}^\top G \frac{\partial H}{\partial u} \leq 0. \tag{4.6}$$

□

As mentioned above, equations of this form are used not only in Hamiltonian mechanics but also in many fields of mathematical modeling, for example, in the Landau theory and the phase-field method. Phenomena such as phase separation, crystal growth, and crack propagation are modeled using these theories (e.g., [9,78,110,121]). The above model also includes the semi-discretized partial differential equations. For example, the KdV equation, which is a model of shallow-water waves,

$$\frac{\partial u}{\partial t} = \alpha u \frac{\partial u}{\partial x} + \beta \frac{\partial^3 u}{\partial x^3},$$

where α, β are parameters. This equation can be written as

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x} \frac{\delta H}{\delta u}, \tag{4.7}$$

where $H(u, \partial u/\partial x)$ is the Hamiltonian and $\delta H/\delta u$ is the variational derivative of H, which is defined by

$$\frac{\delta H}{\delta u} = \frac{\partial H}{\partial u} - \frac{\partial}{\partial x} \frac{\partial H}{\partial u_x},$$

where u_x denotes $\partial u/\partial x$. Equation (4.7) is an example of a Hamiltonian partial differential equation, in which $\partial/\partial x$ in front of $\delta H/\delta u$ plays the role of the skew-symmetric matrix G. In fact, if $\partial/\partial x$ is discretized by using a central difference

operator,

$$\frac{\partial}{\partial x} \simeq D := \frac{1}{2\Delta x} \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & -1 \\ -1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ldots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & -1 & 0 \end{pmatrix}.$$

Because the difference matrix D is skew-symmetric, the semi-discretized equation has the form of equation (4.3) with a skew-symmetric matrix G. Similarly, an equation with the form

$$\frac{\partial u}{\partial t} = \frac{\partial^2}{\partial x^2} \frac{\delta H}{\delta u}, \tag{4.8}$$

is semi-discretized to equation (4.3) with a negative-semidefinite G. In fact, if the $\partial^2/\partial x^2$ in front of $\delta H/\delta u$ is discretized,

$$\frac{\partial^2}{\partial x^2} \simeq D_2 := \frac{1}{\Delta x^2} \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 0 & 1 \\ 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -2 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ldots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -2 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 1 & -2 \end{pmatrix},$$

which is known to be negative semidefinite [33].

The universal approximation properties and the generalization error analysis are also applied to this model in a straightforward way under the assumption that S is non-degenerate. In fact, the properties of the matrix S used in the theoretical analyses are (1) S is non-degenerate and (2) the norm of S is bounded. If S is a constant matrix, (2) is automatically satisfied and hence (1) is the only assumption of the matrix S. In a part of the theoretical analysis, we consider the errors in the gradient ∇H of the energy function. However, typically, the loss function of $S\nabla H$ is minimized in the training process, and if S is degenerate, errors in ∇H may not be small even when those in $S\nabla H$ are small. Hence, S should be non-degenerate.

4.2 Brief Introduction to Hamiltonian Systems and the KAM Theory

We briefly introduce some properties of Hamiltonian systems.

Theorem 4.2 (Darboux)**.** By an appropriate coordinate transformation, the matrix S is transformed into the normal form

$$\begin{pmatrix} O & I \\ -I & O \end{pmatrix}.$$

Definition 4.1. The function $\omega : (v, w) \in \mathbb{R}^N \times \mathbb{R}^N \mapsto \omega(v, w) \in \mathbb{R}$

$$\omega(v, w) = v^\top S^{-1} w$$

is called the symplectic form. Using the symplectic form associates a vector field X_F with each function $F : \mathbb{R}^N \to \mathbb{R}$ by requiring

$$\omega(X_F, w) = \frac{\partial F}{\partial u} \cdot w \qquad \text{for all } w.$$

For two functions F, G, the following operation is called the Poisson bracket:

$$\{F, G\} = \omega(X_F, X_G). \tag{4.9}$$

Definition 4.2. A Hamiltonian system for which the state variable is $N = 2M$ dimensional is integrable in the sense of Liouville if this Hamiltonian system has the first integrals (i.e., conserved quantities) $F_1, F_2, \ldots, F_M$ with $\nabla F_1(u), \nabla F_2(u), \ldots, \nabla F_M(u)$ independent at each u and for all i, j:

$$\{F_i, F_j\} = 0.$$

For integrable systems, Theorem 4.3 is known.

Theorem 4.3 (Liouville–Arnold)**.** Suppose that for an integrable Hamiltonian system, constants $c_1, \ldots, c_M$ exist such that $K = \cap_{i=1}^{M} F_i^{-1}(c_i)$ is connected and compact. Then, there exists a neighborhood $\mathcal{N}$ of K, $\mathcal{U} \subset \mathbb{R}^n$ and a coordinate transform

$$\phi : (\theta, J) \in \mathbb{T}^n \times \mathcal{U} \to \phi(\theta, J) \in \mathcal{N} \tag{4.10}$$

such that the transformed system is the Hamilton equation of which Hamiltonian $H \circ \phi$ depends only on J.

The variables J and θ are called action-angle variables. Theorems 4.2 and 4.3 roughly mean that integrable Hamiltonian systems can be written in the following form:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\theta\\ J\end{pmatrix} = \begin{pmatrix}O & I\\ -I & O\end{pmatrix}\begin{pmatrix}\frac{\partial \tilde{H}}{\partial \theta}\\ \frac{\partial \tilde{H}}{\partial J}\end{pmatrix}.$$

Further, because $\tilde{H} = H \circ \phi$ depends on I only, it holds that

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\theta\\ J\end{pmatrix} = \begin{pmatrix}O & I\\ -I & O\end{pmatrix}\begin{pmatrix}0\\ \frac{\partial \tilde{H}}{\partial J}\end{pmatrix} = \begin{pmatrix}\frac{\partial \tilde{H}}{\partial J}\\ 0\end{pmatrix}.$$

This shows that J is constant, and hence θ moves on the torus at a constant velocity. Because the velocities are typically not co-related to each other, the dynamics are "quasi-periodic." See, for example, [104] for more details.

As seen above, integrable Hamiltonian systems are quasi-periodic. Note that general Hamiltonian systems are not necessarily quasi-periodic and neither are Hamiltonian neural networks. However, for Hamiltonian neural networks that are trained to model integrable systems, the quasi-periodic behaviors are preferably maintained. When the modeling error is sufficiently small, this is considered as a perturbation problem. The perturbation theory of Hamiltonian systems has been investigated from various perspectives. For example, perturbed integrable Hamiltonian systems are in general no longer integrable; hence, approximation of integrable Hamiltonian systems by integrable neural network models appears to be difficult. Fortunately, however, the KAM theory shows that even though the perturbed system is not integrable, it maintains the quasi-periodic behaviors described above under certain conditions.

The KAM theorem has many variants under various conditions. The following variant [104] is typical:

Theorem 4.4 (KAM Theorem)**.** Let θ and J be the action-angle variables for a C^∞ integrable Hamiltonian $H_0 : \mathbb{R}^{2M} \to \mathbb{R}$ with $M \geq 2$. If H_0 is non-degenerate, that is,

$$\det \frac{\partial^2 H_0}{\partial J^2} \neq 0, \tag{4.11}$$

for the perturbed system $H(\theta, J) = H_0(J) + \varepsilon F(\theta, J, \varepsilon)$ by $F \in C^\infty$, there exists ε_0 such that if $\varepsilon F < \varepsilon_0$, there exists a set of M-dimensional tori that are invariant under the perturbed flow. For each invariant torus, the flow of the perturbed system H

is quasi-periodic. In addition, the set of invariant tori is large in the sense that its measure becomes full as $\varepsilon \to 0$.

Remarks. 3. The last sentence – *the set of invariant tori is large in the sense that its measure becomes full as* $\varepsilon \to 0$ – corresponds to the non-existence of so-called *resonance.* If the perturbation added to the system is in resonance with the original system, the perturbation may grow rapidly and the behavior of the system may change significantly. This statement assures that for small perturbations, such a situation almost never occurs.

Remarks. 4. It may be difficult to check whether the target system is integrable by using given data. One possibility is application of the Koopman operator, which makes it possible to find the conserved quantities that the given data may admit. If a sufficient number of conserved quantities exist, it is highly likely that the target system is integrable.

4.3 Main Results

We proved the persistence of the quasi-periodic behaviours of integrable Hamiltonian systems with a high probability even when the loss function is not perfectly zero. Further, we provided a generalization error bound and universal approximation theorems for Hamiltonian neural networks to ensure that the loss function can be sufficiently small for the application of the KAM theorem. In this section we present our main results.

4.3.1 Universal approximation properties of Hamiltonian neural networks

For Hamiltonian neural networks to be close to the true dynamics, a universal approximation theorem and a generalization error analysis are needed. First, we show universal approximation theorems.

We first define some notation to describe the theorem. $C^m(X)$ with the topology of the Sobolev space $W^{p,m}(X)$ is denoted by $S_p^m(X)$, where $W^{p,m}(X)$ is a space of functions that admit weak derivatives up to the mth order that bounds L^p-norms. Hence, $S_p^m(X)$ is the space of functions in $W^{p,m}(X)$ with (usual) derivatives; for

details on the Sobolev theory, see [2]. L^p-norms of functions are denoted by $\|\cdot\|_{L^p}$, and those of vectors by $\|\cdot\|_p$.

Universal approximation theorem for Hamiltonian neural networks The following theorem shows the universal approximation property of Hamiltonian neural networks.

Theorem 4.5. Let $H : \mathbb{R}^N \to \mathbb{R}$ be an energy function with the equation

$$\frac{\mathrm{d}u}{\mathrm{d}t} = S\frac{\partial H}{\partial u},$$

where $u : t \in \mathbb{R} \mapsto u(t) \in \mathbb{R}^N$ and S is a skew-symmetric $N \times N$ matrix. Suppose that the phase space K of this system is compact and the right-hand side $S\partial H/\partial u$ is Lipschitz continuous. If the activation function $\sigma \neq 0$ belongs to $S_2^1(\mathbb{R})$, then for any $\varepsilon > 0$, there exists a neural network H_{NN} for which

$$\left\| S\frac{\partial H}{\partial u} - S\frac{\partial H_{\mathrm{NN}}}{\partial u} \right\|_2 < \varepsilon$$

holds. In addition, if the energy function is C^∞, the function can be approximated by a C^∞ neural network provided that the activation function is sufficiently smooth.

An outline of the proof is as follows. From the existence theorem of a solution to ordinary differential equations, the right-hand side of the equation must be Lipschitz continuous to guarantee the existence of a solution. Therefore, the differential of the H is Lipschitz continuous, which implies the smoothness of H. Then, the approximation property is obtained from the universal approximation theorem for smooth functions by [43]. For the detailed proof is as follow.

Proof of Theorem 4.5 We prove the following universal approximation theorem for the general energy-based physical models.

Theorem 4.6. Let $H : \mathbb{R}^N \to \mathbb{R}$ be an energy function with the equation

$$\frac{\mathrm{d}u}{\mathrm{d}t} = G\frac{\partial H}{\partial u},$$

where $u : t \in \mathbb{R} \mapsto u(t) \in \mathbb{R}^N$ and G is a non-degenerate $N \times N$ matrix. Suppose that the state space K of this system is compact and the right-hand side $G\partial H/\partial u$ is Lipschitz continuous. If the activation function $\sigma \neq 0$ belongs to $S_2^1(\mathbb{R})$, then for any

$\varepsilon > 0$ there exists a neural network H_{NN} for which

$$\left\| G\frac{\partial H}{\partial u} - G\frac{\partial H_{\mathrm{NN}}}{\partial u} \right\|_2 < \varepsilon$$

holds. In addition, if the energy function is C^∞, the function can be approximated by a C^∞ neural network provided that the activation function is sufficiently smooth.

To prove this theorem, we use the following theorem and the lemma, both of which were shown in [43].

Theorem 4.7 (Hornik et al., 1990)**.** Let $\Sigma(\sigma)$ be the space of the neural networks with the activation function σ. If the activation function $\sigma \neq 0$ belongs to $S_p^m(\mathbb{R}, \lambda)$ for an integer $m \geq 0$, then $\Sigma(\sigma)$ is m-uniformly dense in $C^\infty(K)$, where K is any compact subset of $\mathbb{R}^N$.

Lemma 1 (Hornik et al., 1990)**.** Under the same assumption, $\Sigma(\sigma)$ is also dense in $S_p^m(\mathbb{R}, \lambda)$.

Hence, if the activation function σ of the hidden layer is in $S_p^m(\mathbb{R}, \lambda)$ and does not vanish everywhere, then for any sufficiently smooth function, there exists a neural network that approximates the function and its derivatives up to the order m arbitrarily well on compact sets. This theorem has also been extended to the functions of multiple outputs; see [43].

Proof of Theorem 4.5 Because the target equation is determined only by the gradient of H, any function obtained by shifting H by a constant gives the same equation. Hence, we choose and fix an energy function H that yields the target equation. Because $G\partial H/\partial u$ is Lipschitz continuous and hence continuous on the phase space K, this function is bounded and square-integrable. Thus, $G\partial H/\partial u \in S_2^0(K)$, which means H is in $S_2^1(K)$. Therefore, from Lemma 1 and the assumption that the activation function is in $S_2^1(\mathbb{R})$, for each ε, there exists a neural network that approximates H in $S_2^1(K)$:

$$\|H - H_{\mathrm{NN}}\|_2^2 + \left\| \frac{\partial H}{\partial u} - \frac{\partial H_{\mathrm{NN}}}{\partial u} \right\|_2^2 < \frac{\varepsilon^2}{\|G\|_2^2},$$

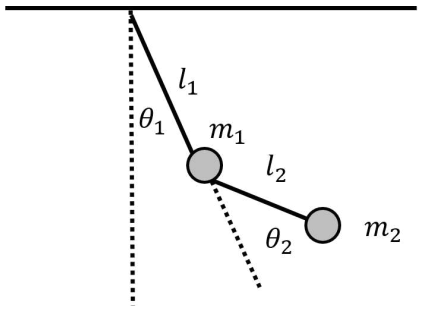


Fig.4.2 Double pendulum used as the target in the experiment for the illustration of the model with the coordinate transformations.

which gives

$$\left\| G\frac{\partial H}{\partial u} - G\frac{\partial H_{\mathrm{NN}}}{\partial u} \right\|_2^2 \leq \|G\|_2^2 \left\| \frac{\partial H}{\partial u} - \frac{\partial H_{\mathrm{NN}}}{\partial u} \right\|_2^2 < \varepsilon^2.$$

□

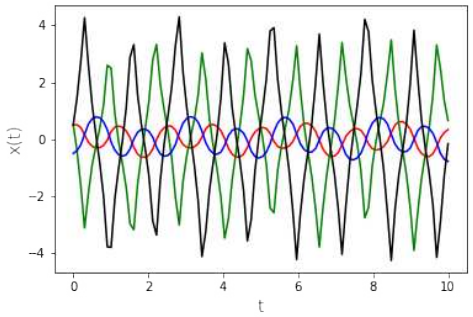
Hamiltonian neural networks with a coordinate transformation The practical use of Hamiltonian neural networks is hampered by the fact that the state variables must be represented by a specific coordinate, such as the generalized momentum; however, the derivation of the generalized momentum requires the energy function, which is unknown. For example, the double pendulum in Fig. 4.2 exhibits the dynamics shown in Fig. 4.3. These are predicted by the models that are trained from the data of the state variables and their derivatives, not those of the generalized momenta. Hamiltonian neural networks failed to solve such problems because the data were not given in the canonical coordinate system. For the details, see Section 3.1.

Based on this, we here propose a model with a coordinate transformation, such as the transformations that appear, for example, in [50, 94].

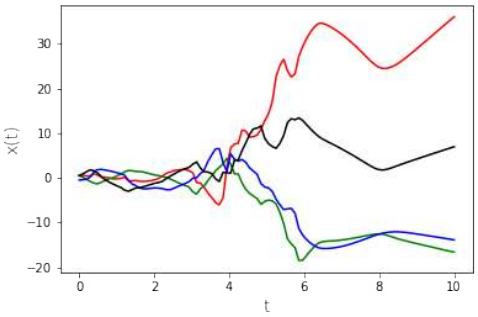
Suppose that, although the given data $x(t)$ is not represented by the canonical coordinate system, the data $x(t)$ can be transformed into the canonical coordinate system by an unknown transformation $u(t) = u_{\mathrm{NN}}[x(t)]$. By substituting $u = u_{\mathrm{NN}}(x)$ into the model equation (4.2), we obtain

$$\frac{\mathrm{d}x}{\mathrm{d}t} = (\frac{\partial u_{\mathrm{NN}}}{\partial x})^{-1} S (\frac{\partial u_{\mathrm{NN}}}{\partial x})^{-\top} \frac{\partial H_{\mathrm{NN}}}{\partial x}. \tag{4.12}$$

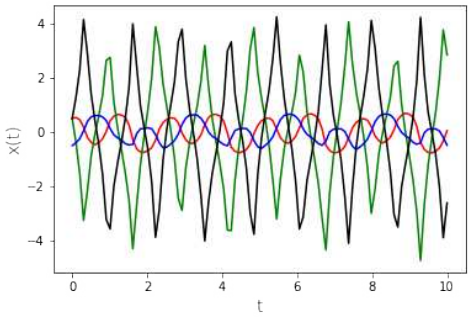
We show that model (4.12) admits the same energetic property as the original equation and also the universal approximation property.



(a) Ground truth



(b) Naive HNN model



(c) HNN model with a coordinate transformation

Fig.4.3 Examples of the orbits predicted by a Hamiltonian neural network and the model with coordinate transformations. Each component of $x(t) = [q_1(t), v_1(t), q_2(t), v_2(t)]$ is represented as red (q_1), green (v_1), blue (q_2), and black (v_2).

Theorem 4.8. The model (4.12) admits the energy conservation law in the sense that $\mathrm{d}H_{\mathrm{NN}}/\mathrm{d}t = 0$.

Proof. By substituting the equation, we obtain

$$\frac{\mathrm{d}H_{\mathrm{NN}}}{\mathrm{d}t} = \frac{\partial H_{\mathrm{NN}}}{\partial x}^{\top} \frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial H_{\mathrm{NN}}}{\partial x}^{\top} \frac{\partial u_{\mathrm{NN}}}{\partial x}^{-1} S \frac{\partial u_{\mathrm{NN}}}{\partial x}^{-\top} \frac{\partial H_{\mathrm{NN}}}{\partial x} = 0$$

because S is skew-symmetric and hence for any vector v, $v^{\top} S v = 0$. □

Theorem 4.9. Let $H : \mathbb{R}^N \to \mathbb{R}$ be an energy function for the equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = (\frac{\partial u}{\partial x})^{-1} S (\frac{\partial u}{\partial x})^{-\top} \frac{\partial H}{\partial x},$$

where $x : t \in \mathbb{R} \mapsto x(t) \in \mathbb{R}^N$, $u : x \in \mathbb{R}^N \mapsto u(x) \in \mathbb{R}^N$, and S is an $N \times N$ matrix. Suppose that the phase space K of this system is compact and the right-hand side $\partial H/\partial u$ is Lipschitz continuous. Suppose also that u is a C^1-diffeomorphism. If the functions $\sigma \neq 0$ and $\rho \neq 0$ belong to $S_2^1(\mathbb{R})$, then for any $\varepsilon > 0$, there exist neural networks H_{NN} with the activation functions σ and u_{NN} with ρ for which

$$\left\| (\frac{\partial u}{\partial x})^{-1} S (\frac{\partial u}{\partial x})^{-\top} \frac{\partial H}{\partial x} - (\frac{\partial u_{\mathrm{NN}}}{\partial x})^{-1} S (\frac{\partial u_{\mathrm{NN}}}{\partial x})^{-\top} \frac{\partial H_{\mathrm{NN}}}{\partial x} \right\|_2 < \varepsilon$$

holds.

We prove the following theorem, which is a generalization of Theorem 4.9.

Theorem 4.10. Let $H : \mathbb{R}^N \to \mathbb{R}$ be an energy function for the equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = (\frac{\partial u}{\partial x})^{-1} G (\frac{\partial u}{\partial x})^{-\top} \frac{\partial H}{\partial x},$$

where $x : t \in \mathbb{R} \mapsto x(t) \in \mathbb{R}^N$, $u : x \in \mathbb{R}^N \mapsto u(x) \in \mathbb{R}^N$, and S is an $N \times N$ non-degenerate matrix. Suppose that the phase space K of this system is compact and the right-hand side $\partial H/\partial u$ is Lipschitz continuous. Suppose also that u is a C^1-diffeomorphism. If the functions $\sigma \neq 0$ and $\rho \neq 0$ belong to $S_2^1(\mathbb{R})$, then for any $\varepsilon > 0$ there exist neural networks H_{NN} with the activation functions σ and u_{NN} with ρ for which

$$\left\| (\frac{\partial u}{\partial x})^{-1} G (\frac{\partial u}{\partial x})^{-\top} \frac{\partial H}{\partial x} - (\frac{\partial u_{\mathrm{NN}}}{\partial x})^{-1} G (\frac{\partial u_{\mathrm{NN}}}{\partial x})^{-\top} \frac{\partial H_{\mathrm{NN}}}{\partial x} \right\|_2 < \varepsilon$$

holds.

Proof. We need to prove the approximation property for $(\partial u/\partial x)^{-1}$. From the assumption that $\rho \neq 0$ is in $S_2^1(\mathbb{R})$, there exists a function u_{NN} that approximates $\partial u/\partial x$. Because the determinant function of matrices is continuous, it is deduced that $\det \partial u_{\mathrm{NN}}/\partial x \neq 0$ and hence $(\partial u_{\mathrm{NN}}/\partial x)^{-1}$ exists. Because the matrix inverse is also continuous, $(\partial u_{\mathrm{NN}}/\partial x)^{-1}$ is also approximated by u_{NN}. □

4.3.2 Generalization error analysis of Hamiltonian neural networks

Next, we derive a generalization error bound for the standard Hamiltonian neural network (4.2) by employing a technique from statistical learning theory. More precisely, we adjust the technique so that an estimate on the energy gradient can be obtained.

Remarks. 5. Although the standard Hamiltonian neural network without the coordinate transformations is considered below, the results can be extended to the general energy-based model with the coordinate transformations if the matrix $(\partial u/\partial x)^{-1}$ is bounded.

In statistical learning theory, generalization error bounds are typically obtained by using the Rademacher complexities. See, for example, [8, 36, 106, 111] for details.

Definition 4.3. For a set $V \subset \mathbb{R}^n$,

$$\mathcal{R}_n(V) := \frac{1}{n}\mathbb{E}_{\sigma\sim\{-1,1\}^n} \sup_{v\in V} \sum_{i=1}^{n} \sigma_i v_i$$

is called the Rademacher complexity of V.

Lemma 2. Let X and Y be arbitrary spaces, $\mathcal{F} \subset \{f : X \to Y\}$ be a hypotheses class, and $L : Y \times Y \to [0, c]$ be a loss function. For a given data set $(x_i, y_i) \in X \times Y$ $(i = 1, \ldots, n)$, let $\mathcal{G}$ be defined by $\{(x_i, y_i) \in X \times Y \mapsto L[y_i, h(x_i)] \in \mathbb{R} \mid h \in \mathcal{F}, i = 1, \ldots, n\}$. Then, for any $\delta > 0$ and any probability measure P, we obtain, with a probability of at least $(1-\delta)$ with respect to the repeated sampling of P^n-distributed training data, the following:

$$E[L(Y, h(X))] \leq \frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i)) + 2\mathcal{R}_n(\mathcal{G}) + 3c\sqrt{\frac{2\ln\frac{4}{\delta}}{n}}$$

for all $h \in \mathcal{F}$.

The Rademacher complexity is known to be bounded by using the covering number.

Definition 4.4. Let V and V' be subsets of $\mathbb{R}^n$. V is r-covered by V' with respect to the metric function defined by the norm $\|\cdot\|$ if for all $v \in V$, there exists a $v' \in V'$ such that $\|v - v'\| < r$. The covering number $N(r, V, \|\cdot\|)$ of V is the minimum

number of elements of a set that r covers V. $N(r, V, \|\cdot\|)$ is also denoted by $N(r, V)$ if the metric is clear from the context.

Lemma 3. If $\sqrt{\log N(c2^{-k}, V)} \leq \alpha + k\beta$ for some α and β, then $\mathcal{R}_n(V) \leq 6c(\alpha + 2\beta)/n$.

Thus, if the covering number is estimated for a Hamiltonian neural network, the bound on the generalization error is obtained. To this end, we suppose that the model is trained by minimizing the p-norm of the error in the right-hand side of the model. More precisely, for the hypothesis $h : u_j \mapsto S\frac{\partial H_{\mathrm{NN}}(u_j)}{\partial u}$, we consider the loss function

$$L[\nabla H(u_j), h(u_j)] = \left\| \frac{\partial H(u_j)}{\partial u} - \frac{\partial H_{\mathrm{NN}}(u_j)}{\partial u} \right\|_p^p, \tag{4.13}$$

where u_i are training data. We denote the Lipschitz constant of the loss function associated with the above by ρ_p. Of course, $p = 2$ is typically used; however, we show below that $p > 2M$ is useful to obtain an L^∞ bound on the Hamiltonian.

Remarks. 6. The training can be performed also by using the symplectic gradient:

$$\left\| S\frac{\partial H(u_j)}{\partial u} - S\frac{\partial H_{\mathrm{NN}}(u_j)}{\partial u} \right\|_p^p. \tag{4.14}$$

In that case, the results will be slightly modified using the norm of S; however, we omit this for simplicity.

A bound of the covering number is derived as follows.

Theorem 4.11. Suppose that the hypotheses class $\mathcal{F}$ consists of multi-layer perceptrons f_{NN} that have ρ_{σ_j}-Lispchitz activation functions $\sigma_j (j = 1, \ldots, n_l)$, for which the derivatives are ρ'_j-Lipschitz continuous and bounded by $\sup|\sigma'_j| < c_{\sigma_j}$. Suppose also that the matrices $A_j^\top (j = 1, \ldots, n_l + 1)$ in the linear layers in the perceptrons have the bounded norm $|A_j^\top| < c_{A_j}$:

$$\mathcal{F} = \{f_{\mathrm{NN}}(u) \mid A_{n_l+1}\sigma_{n_l}(A_{n_l}\sigma_{n_{l-1}}[\cdots\sigma_1(A_1u + b_1)\cdots] + b_{n_l}) + b_{n_l+1}\},$$

where b_j's are vectors. Let $\mathcal{G}$ be defined by $\{L[\nabla H(u_i), h(u_i)] \mid h \in \mathcal{F}\}$ with the ρ_p-Lipschitz continuous loss function L. In addition, suppose that the phase space is compact so that the data $u_i (i = 1, \ldots, n)$ are in a bounded set with the bound

$\|u_i\| < c_u$. Then, the covering number of $\mathcal{G}$ is estimated by

$$N(\varepsilon, \mathcal{G}) \leq \Big(\frac{2\rho_p c_u c_{A_{n_{l+1}}} \rho'_{\sigma_{n_l}} (\prod_{j=1}^{n_l - 1} \rho_{\sigma_j})(\prod_{j=1}^{n_l - 1} c_{\sigma_j})(\prod_{j=1}^{n_l} c_{A_j})^2}{\varepsilon} + 1 \Big)^n.$$

To prove this theorem, we use the following lemmas, which are typically used to estimate the covering numbers [106].

Lemma 4. Let B be a unit ball in $\mathbb{R}^n$. Then, $N(\varepsilon, B, \|\cdot\|_2) \leq \left(\frac{2}{\varepsilon} + 1\right)^n$.

Lemma 5. Suppose that functions $\phi_i : \mathbb{R} \to \mathbb{R}$, $i = 1, 2, \ldots, n$ are ρ-Lipschitz continuous. Then, for $V \subset V^n$, $N(\varepsilon, \vec{\phi} \circ V) \leq N(\frac{\varepsilon}{\rho}, V)$, where for $v \in \mathbb{R}^n$, $\vec{\phi}(v) := [\phi_1(v_1), \ldots, \phi_n(v_n)]$, $\vec{\phi} \circ V := \{\vec{\phi}(v) \mid v \in V\}$.

Proof of Theorem 4.11 To simplify the discussion, we will estimate the covering number of the following perceptron

$$f_{\mathrm{NN}}(u) = A_3 \sigma_2 [A_2 \sigma_1 (A_1 u + b_1) + b_2] + b_3.$$

Because the proof for general cases is exactly the same, we need to estimate the covering number of the gradient of f_{NN}, which is written as

$$\nabla f_{\mathrm{NN}}(u) = A_1^\top (D\sigma_1) A_2^\top (D\sigma_2) A_3^\top,$$

where $D\sigma_2$ and $D\sigma_1$ are Jacobian matrices. These Jacobian matrices are evaluated at $u = A_2 \sigma_1 (A_1 u + b_1) + b_2$ and $A_1 u + b_1$, respectively. We first estimate the covering number associated with $D\sigma_2$. $D\sigma_2$ has the same architecture as a multi-layer perceptron, except that the last activation function is replaced by the differential σ_2' of σ_2:

$$D\sigma_2 = \sigma_2' [A_2 \sigma_1 (A_1 u + b_1) + b_2].$$

Assuming that the input data are in the ball B_{c_u} with radius c_u, we obtain

$$N(\varepsilon, B_{c_u}) \leq \left(\frac{2c_u}{\varepsilon} + 1 \right)^n.$$

Then, because the norms of A_1, A_2 are bounded by c_{A_1}, c_{A_2}, matrix multiplications by these matrices are c_{A_1}- and c_{A_2}-Lipschitz continuous, respectively. In addition,

σ_1 is ρ_1-Lipschitz and σ_2' is ρ_2'-Lipschitz continuous. Therefore, the covering number associated with $D\sigma_2$ is estimated by

$$N(\varepsilon, D\sigma_2) \leq \left(\frac{2\rho_1\rho_2' c_{A_1} c_{A_2} c_u}{\varepsilon} + 1\right)^n.$$

Finally, because σ_1' is assumed to be bounded by c_{σ_1}, the norms of the matrices other than $D\sigma_2$ in ∇f_{NN} are bounded as follows: $\|A_1^\top\| < c_{A_1},\ \|A_2^\top\| < c_{A_2},\ \|A_3^\top\| < c_{A_3},\ \|D\sigma_1\| < c_{\sigma_1}$. Because the loss function is assumed to be ρ_p-Lipschitz continuous, we obtain the estimate

$$N(\varepsilon, \mathcal{G}) \leq \left(\frac{2\rho_p\rho_1\rho_2' c_{\sigma_1}(c_{A_1}c_{A_2})^2 c_{A_3} c_u}{\varepsilon} + 1\right)^n.$$

□

4.3.3 L^∞ estimate on the error in the Hamiltonian

The generalization error analysis in Theorem 4.11 shows that, at a certain probability, the expectation of the loss function can be bounded. If this bound certainty holds and if the training is performed by minimizing the p-norm with $p > 2M$, we can derive an L^∞ estimate on the Hamiltonian for the standard Hamiltonian neural network (4.2) by applying the Poincaré inequality and the Sobolev inequality under Assumption 1.

Assumption 1 There exists a density f_P for measure P with $\inf f_P > 0$.

Remarks. 7. The condition $p > 2M$ is not required in practice because of the well-known equivalence of the norms in finite dimensional spaces; for example, if the standard 2-norm is small enough, then the p-norm is also small. However, when the dimension $2M$ is large, the 2-norm needs to be very small to bound the p-norm because the constant in the inequality used to bound the p-norm depends on the dimension. Therefore, it is preferable to minimize the p-norm in such cases.

Theorem 4.12 (Poincaré inequality)**.** Suppose that $1 \leq p \leq \infty$ and $\Omega \subset \mathbb{R}^{2M}$ is bounded. Then there exists a constant c_{p} such that, for any $H \in S_p^1(\Omega)$,

$$\int_\Omega |H(u) - \bar{H}|^p \mathrm{d}u \leq c_{\mathrm{p}} \left\| \frac{\partial H}{\partial u} \right\|_p^p,\ \bar{H} = \frac{1}{\int_\Omega \mathrm{d}u} \int_\Omega H(u)\mathrm{d}u.$$

The constant c_{p} is called the Poincaré constant.

Theorem 4.13 (Sobolev inequalities)**.** There exist constants c_1, c_2 such that, if $lp > 2M$,

$$\|e\|_{L^\infty}(\mathbb{R}^{2M}) \leq c\|e\|_{W^{p,l}}(\mathbb{R}^{2M}), \quad \|e\|_{L^\infty}(\mathbb{T}^{2M}) \leq c\|e\|_{W^{p,l}}(\mathbb{T}^{2M}).$$

By using these inequalities along with the invariance of the Hamilton equation under the constant shift of the energy function, we obtain an error bound on the Hamiltonian.

Lemma 6. Among the energy functions that yield the target Hamilton equation, we choose the one for which

$$\int H(u)\mathrm{d}u = \int H_{\mathrm{NN}}(u)\mathrm{d}u \tag{4.15}$$

holds, so that the error function has zero mean: $e(u) := H(u) - H_{\mathrm{NN}}(u),\ \bar{e}(u) := 0$. Then,

$$\int_\Omega |e(u)|^p \mathrm{d}u \leq c_{\mathrm{p}} \left\| \frac{\partial e}{\partial u} \right\|_{L^p}^p.$$

From the above estimate, we obtain

$$\int \left\| \frac{\partial H(u)}{\partial u} - \frac{\partial H_{\mathrm{NN}}(u)}{\partial u} \right\|_p^p \mathrm{d}P \leq \frac{1}{n}\sum_{i=1}^n L[Y_i, h(X_i)] + 2\mathcal{R}_n(\mathcal{G}) + 3c\sqrt{\frac{2\ln\frac{4}{\delta}}{n}}.$$

By using the density f_P for the measure P, we obtain

$$\inf f_P \int \left\| \frac{\partial H(u)}{\partial u} - \frac{\partial H_{\mathrm{NN}}(u)}{\partial u} \right\|_p^p \mathrm{d}u \leq \int \left\| \frac{\partial H(u)}{\partial u} - \frac{\partial H_{\mathrm{NN}}(u)}{\partial u} \right\|_p^p \mathrm{d}P,$$

which gives us

$$\begin{aligned}
&\int \left\| \frac{\partial H(u)}{\partial u} - \frac{\partial H_{\mathrm{NN}}(u)}{\partial u} \right\|_p^p \mathrm{d}u \\
&\leq \frac{1}{\inf f_P} \int \left\| \frac{\partial H(u)}{\partial u} - \frac{\partial H_{\mathrm{NN}}(u)}{\partial u} \right\|_p^p \mathrm{d}P \\
&\leq \frac{1}{\inf f_P} \left(\frac{1}{n}\sum_{i=1}^n L[Y_i, h(X_i)] + 2\mathcal{R}_n(\mathcal{G}) + 3c\sqrt{\frac{2\ln\frac{4}{\delta}}{n}} \right).
\end{aligned}$$

We note that the left-hand side is the Sobolev norm of the error in $W^{p,l}$; then, under the assumption that $p > 2M$, we can use the Sobolev inequality to obtain

$$(\sup_u \|H(u) - H_{\mathrm{NN}}(u)\|)^p \leq c^p \left\| \frac{\partial H(u)}{\partial u} - \frac{\partial H_{\mathrm{NN}}(u)}{\partial u} \right\|_p^p$$
$$\leq \frac{c^p}{\inf f_P} \left(\frac{1}{n} \sum_{i=1}^{n} L[Y_i, h(X_i)] + 2\mathcal{R}_n(\mathcal{G}) + 3c\sqrt{\frac{2\ln \frac{4}{\delta}}{n}} \right),$$

which ensure that H and H_{NN} are close in terms of the function values.

4.3.4 KAM theory for Hamiltonian neural networks

The universal approximation property shown in the previous sections guarantees that the value of MSE can be made arbitrarily small by training; however, in actual training, a finite error remains. In this section, as an application of the generalization bound, we apply the KAM theory to theoretically investigate the trained standard Hamiltonian neural network model (4.2) in such cases by assuming that the target system is integrable.

We make a few assumptions that are needed for the application of the KAM theory.
Assumption 2 The dimension of the phase space is assumed to be $2M$ with $M \geq 2$.
Assumption 3 The target system is an integrable Hamiltonian system with the conserved quantities $F_1, \ldots, F_M$. The series $c_1, \ldots, c_M$ exists such that $K = \cap_{i=1}^{M} F_i^{-1}(c_i)$ is connected and compact.

Under the above assumptions, from the Liouville–Arnold theorem there exist a neighborhood $\mathcal{N}$ of K, $\mathcal{U} \subset \mathbb{R}^n$ and a coordinate transform

$$\phi : (\theta, J) \in \mathbb{T}^n \times \mathcal{U} \to \phi(\theta, J) \in \mathcal{N}, \tag{4.16}$$

such that the transformed system is the Hamilton equation. Following the usual setting of the KAM theorem, we consider the target system and the Hamiltonian equation in the transformed coordinate $\mathbb{T}^n \times \mathcal{U}$.
Assumption 4 The Hamiltonian $H : \mathbb{T}^n \times \mathcal{U} \to \mathbb{R}$ of the target system is C^∞ and non-degenerate. The activation functions of the Hamiltonian neural network used are in C^∞.
Assumption 5 From the generalization error analysis in the previous section, we

have essentially shown that if $p > 2M$, with at least probability $1 - \delta$, it holds that

$$\sup |H(u) - H_{\mathrm{NN}}(u)| < c_1 L_{\mathrm{train}} + c_2 R_n + c_3 \sqrt{\frac{\ln \frac{1}{\delta}}{n}}$$

with constants c_1, c_2, and c_3, where R_n is a bound on the Rademacher complexity. We assume that the training was performed with $p > 2M$ and the above statement certainly holds.

Using these assumptions, we obtain Theorem 4.14.

Theorem 4.14. Let the threshold of the KAM theorem be ε_0 and δ be

$$\delta = \exp\left(-n\left(\frac{\varepsilon_0 - c_1 L_{\mathrm{train}} - c_2 R_n}{c_3}\right)^2\right).$$

Under the above assumptions, with a probability of at least $(1 - \delta)$, a set of invariant tori exists for the trained model H_{NN}.

Proof. It is confirmed by a straightforward calculation that if δ is given as described above, it holds that $\sup |H(u) - H_{\mathrm{NN}}(u)| < \varepsilon_0$, and hence the assumption of the KAM theorem is satisfied. □

Remarks. 8. As mentioned in Remark 1, the KAM theorem also shows that the invariant tori become larger when the perturbation becomes smaller. Hence, if the generalization error is small enough, the size of the tori is expected to be large.

Note that general Hamiltonian systems, and hence general Hamiltonian neural networks, are not quasi-periodic. Therefore, a model that approximates a quasi-periodic Hamilton equation may be (in some sense) *approximately* quasi-periodic, but it is not necessarily *strictly* quasi-periodic. This theorem states that *the trained model can be strictly quasi-periodic even if the training loss does not completely vanish.*

Numerical Example: Learning the Zabusky and Kruskal Experiment As a numerical experiment, we trained a Hamiltonian neural network[*1] so that the dynamics of the KdV equation is learned by using the data from the experiment by [133], in which a nontrivial recurrence of initial states is reported.

[*1] We use the Hamiltonian neural network code for the KdV equation provided by `https://github.com/tksmatsubara/discrete-autograd` (MIT License).

The KdV equation is derived from the following energy function H:

$$H(u) = \int \left[\frac{1}{6}\alpha u^3 - \frac{1}{2}\beta \left(\frac{\partial u}{\partial x} \right)^2 \right] \mathrm{d}x.$$

In fact, under the periodic boundary condition, the variational derivative is

$$\frac{\delta H}{\delta u} = \int \left[\frac{1}{2}\alpha u^2 + \beta \frac{\partial^2 u}{\partial x^2} \right] \mathrm{d}x.$$

Then, the time evolution is expressed as a Hamiltonian equation:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\frac{\delta H}{\delta u} = \alpha u \frac{\partial u}{\partial x} + \beta \frac{\partial^3 u}{\partial x^3}.$$

For spatial discretization, we used the forward and backward difference operators,

$$D_\mathrm{f} := \frac{1}{\Delta x} \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & -1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ldots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & -1 \end{pmatrix} \text{ and}$$

$$D_\mathrm{b} := \frac{1}{\Delta x} \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & -1 \\ -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ldots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix},$$

respectively. The central difference operator D is their mean, specifically $D = \frac{1}{2}(D_\mathrm{f} + D_\mathrm{b})$ and that for the second derivative is $D_2 = D_\mathrm{f} D_\mathrm{b} = D_\mathrm{b} D_\mathrm{f}$. Using these difference operators, the equation is semi-discretized as

$$\begin{aligned} H(u) &= \sum_x \left[\frac{1}{6}\alpha u^3 - \frac{1}{2}\beta \frac{(D_\mathrm{f} u)^2 + (D_\mathrm{b} u)^2}{2} \right] \Delta x, \\ \frac{\mathrm{d}u}{\mathrm{d}t} &= D\frac{\partial H}{\partial u} = D\left(\frac{1}{2}\alpha u^2 + \beta D u \right). \end{aligned}$$

Following [133], we set the parameters to $\alpha = -1.0$ and $\beta = -0.022^2$, set the width of phase space to 2.0, and used the initial condition $u(0, x)$ to $u(0, x) = \cos(x\pi)$. We

discretized the system with the spatial and temporal mesh sizes of $\Delta x = 0.1$ and $\Delta t = 0.01$. We obtained an orbit for 200 time steps from the initial condition using the fifth-order Dormand–Prince method with the absolute and relative tolerances of 10^{-10} and 10^{-8}.

We performed the experiments on an NVIDIA TITAN V with double precision. We employed a three-layered convolutional neural network with kernel sizes of 3, 1, and 1. The number of hidden channels was 200, the number of output channels was 1, the activation function was the tanh function, and each weight parameter was initialized as a random orthogonal matrix. We summed up the output in the spatial direction and obtained the global energy. We used the whole orbit at every iteration, and minimized the mean squared error of the time derivative as the loss function using the Adam optimizer with a learning rate of 10^{-3} for 10,000 iterations; the error reached a maximum of 1.37×10^{-3}. Given the true dynamics u, the absolute error between the energy function H and the neural network H_{NN} was 1.31×10^{-4} on average and 2.51×10^{-4} at most.

Using the true model and the trained neural network, we also obtained orbits for 1100 time steps from the same initial condition, as shown in the second and third panels of Fig. 4.4, respectively. In the top panels, blue and orange lines denote the true state u and the state predicted by the trained neural network u_{NN} at t =0.0, 2.0, 4.8, and 9.8. The bottom panel shows the energy function H given the predicted states u and u_{NN}. Due to the non-zero training error, more waves incur a larger error. Nonetheless, at around $t = 9.8$, the true model and learned neural network reproduce sin waves, which are given as the initial condition, and the energy error is restored to zero; they exhibit quasi-periodic behaviors.

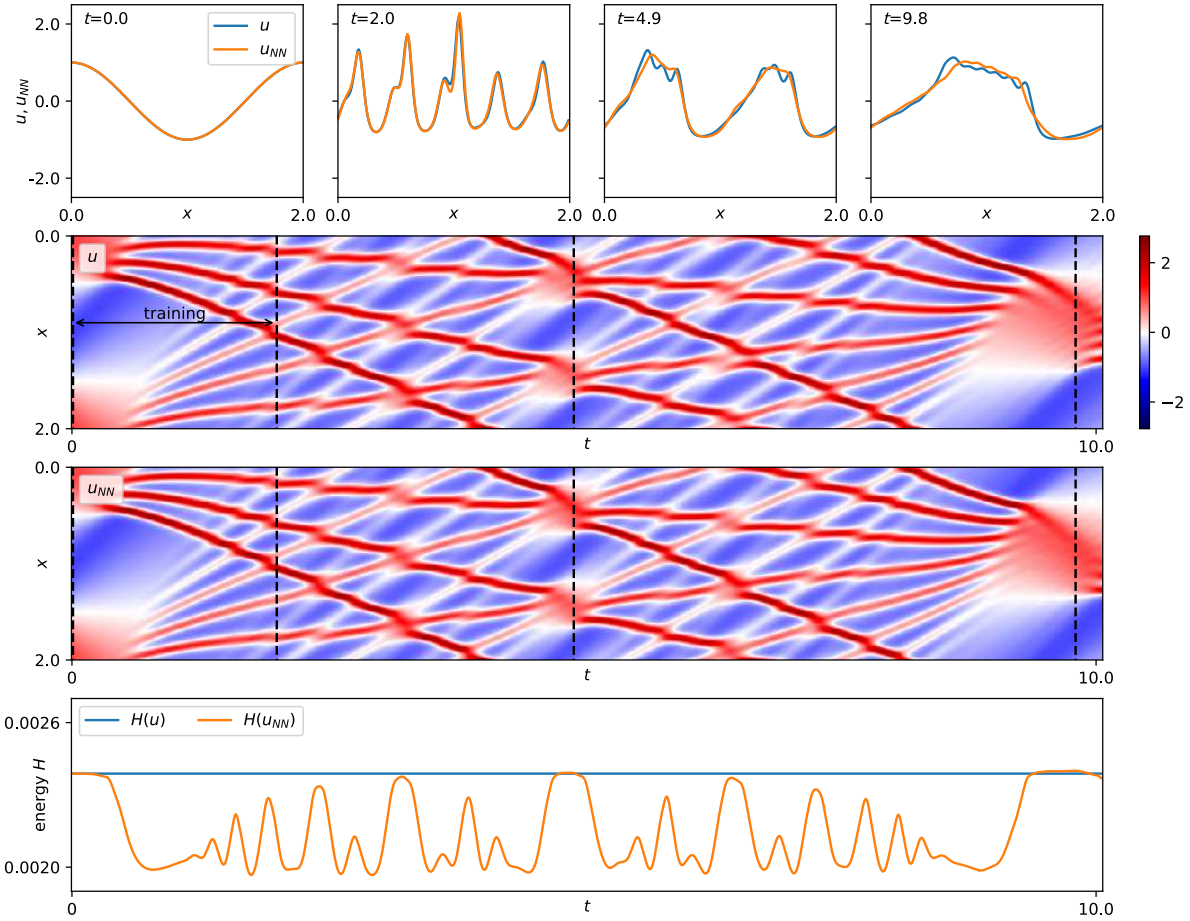


Fig.4.4 Results of training in the Zabusky and Kruskal experiment [133]. (top panels) The predicted states at t =0.0, 2.0, 4.8, and 9.8. (second panel) The true dynamics u. (third panel) The dynamics u_{NN} modeled by a neural network. (bottom panel) The energy function H given the true dynamics u and modeled dynamics u_{NN}.

Chapter 5 Variational Integrator for Hamiltonian Neural Networks and Neural Symplectic Forms

A primal application of the Hamiltonian neural network and the neural symplectic form we proposed in Chapter 3 are physical simulations, and to perform the simulations, these models must be discretized. In this chapter, we present variational integrators for discretization of Hamiltonian neural networks and neural symplectic forms, respectively. First, we consider Hamiltonian neural networks for the following Hamilton equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} O & I \\ -I & O \end{pmatrix} \nabla H. \tag{5.1}$$

Traditional numerical integrators destroy physical properties, such as the energy conservation law. Numerical methods that preserve physical properties are called structure-preserving numerical integrators; however, these are designed for, typically, the Hamilton equations with known Hamiltonian, and hence, application to deep physical models may not be straightforward. Hence it is necessary to develop numerical methods that can be applied to deep physical models. Typical integrators are symplectic integrators, which can be derived as variational integrators [74]. By discretizing the variational principle, the variational integrator discretizes the Euler–Lagrange equation while preserving various conservation laws. Thus one of the aims of this study is to propose a variational integrator for Hamiltonian neural networks.

To address our problems, we need to solve the following two questions.

- Is it possible to formulate a discrete variational problem similar to the variational problem for the Hamilton equation using the energy function given by a neural network?

- Can the discrete Hamilton equation be derived from the formulated discrete variational problem?

In particular, regarding the second question, because the Hamiltonian is provided by a neural network, it is not possible to execute variational calculus manually. Therefore, perhaps it is necessary to apply automatic differentiation to perform the discrete variational calculus.

Meanwhile, most symplectic integrators are designed for the Hamilton equation of the canonical form, and they may not be available for the neural symplectic forms because the Hamilton equations of the neural symplectic forms are not canonical. In this study, we also show that the variational integrator is available for neural symplectic forms, that is, the model of neural symplectic form admits a variational principle that can be used to derive variational integrators.

In addition, the energy conservation property of the proposed method was confirmed by the numerical experiment.

The main contributions of this study include:

1. Physical simulations while preserving the physical properties just by giving the data. The proposed variational integrators enable physical simulations using the Hamiltonian neural network and neural symplectic form without destroying the physical properties, such as the energy conservation law. Further, our results enable users to perform physical simulations just by supplying the observed data in an arbitrary coordinate system while preserving physical properties.

2. Designing various symplectic integrators. The variational integrator can derive a variety of symplectic integrators, depending on the discretization method of the action integral. Hence, numerical integrators that meet the needs of the application, such as accuracy and computational complexity, can be designed.

This chapter is organized as follows. First, in Section 5.1, we briefly introduce the basics of the varitaional integrator. In Section 5.2, we explain varitaional integrators for Hamiltonian neural networks and conduct numerical experiments. In Section 5.3, we show that the variational principle that derives the neural symplectic form certainly exists, and develop a variational integrator for this model.

5.1 Outline of the Variational Integrator

The variational integrator was proposed as a discretization method for the Euler–Lagrange equation [74] , which is the fundamental equation of Lagrangian mechanics, by using the variational principle. First, the variational principle and the Euler–Lagrange equation are explained briefly. Let $q(t) : t \in \mathbb{R} \mapsto q(t) \in \mathbb{R}^n$ denote a variable that represents the state. Consider the case of a mass point in motion under a force derived by a potential energy $V(q)$. The Lagrangian $\mathcal{L}(q, \dot{q})$, which depends on q and its time derivative $\dot{q}$, is defined as the difference between the kinetic and the potential energy:

$$\mathcal{L}(q, \dot{q}) := \frac{m}{2}\dot{q} \cdot \dot{q} - V(q),$$

where the mass is denoted by m. The Euler–Lagrange equation, which is the equation of motion in Lagrangian mechanics, is defined as follows:

$$\frac{\partial \mathcal{L}}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}} = 0. \tag{5.2}$$

This equation is known to be equivalent to Newton's equation of motion. The variational principle states that the Euler–Lagrange equation is obtained by computing the stationary points of the action integral S, which is defined as

$$S := \int_0^T \mathcal{L}(q, \dot{q})\mathrm{d}t.$$

The variational integrator uses this principle for discretization. More precisely, in general, numerical integrators of differential equations are derived by discretizing a given differential equation; however, to obtain the variational integrator, the variational principle is discretized, that is, the action integral is discretized and stationary points of the discretized action integral are computed. In the following, the approximate value of $q(n\Delta t)$ is denoted by $q^{(n)}$, where Δt is the time step size. Then, $\dot{q}$ can be approximated, for example, as follows:

$$\dot{q} \simeq \frac{q^{(n+1)} - q^{(n)}}{\Delta t}.$$

Suppose that the Lagrangian is given as (5.2). Then the Lagrangian can be approximated by

$$\mathcal{L}_{\mathrm{d}}(q^{(n)}, q^{(n+1)}) := \frac{m}{2}\frac{q^{(n+1)} - q^{(n)}}{\Delta t} \cdot \frac{q^{(n+1)} - q^{(n)}}{\Delta t} - V(q^{(n)}).$$

Using this discretized Lagrangian, the action sum S_d is defined as follows:

$$S_\mathrm{d} = \sum_{n=0}^{N-1} \mathcal{L}_\mathrm{d}(q^{(n)}, q^{(n+1)})\Delta t.$$

It is easy to confirm that this is an approximation of the action integral S. In the variational integrator, as in ordinary Lagrangian mechanics, the discrete equations of motion are derived by computing the variation of the action sum. Let $\delta q^{(n)}$ be a variation with $q^{(n)}$ with both ends fixed

$$\delta q^{(0)} = \delta q^{(N)} = 0$$

as is common in the original variational principle. Ignoring higher-order terms, the computation of difference of S_d leads to

$$\begin{aligned}
&\sum_{n=0}^{N-1} \mathcal{L}_\mathrm{d}(q^{(n)} + \delta q^{(n)}, q^{(n+1)} + \delta q^{(n+1)})\Delta t - \sum_{n=1}^{N} \mathcal{L}_\mathrm{d}(q^{(n)}, q^{(n+1)})\Delta t \\
&= \sum_{n=0}^{N-1} \left(D_1\mathcal{L}_\mathrm{d}(q^{(n)}, q^{(n+1)})\delta q^{(n)} + D_2\mathcal{L}_\mathrm{d}(q^{(n)}, q^{(n+1)})\delta q^{(n+1)} \right) \Delta t \\
&= \sum_{n=0}^{N-1} \left(D_1\mathcal{L}_\mathrm{d}(q^{(n)}, q^{(n+1)}) + D_2\mathcal{L}_\mathrm{d}(q^{(n-1)}, q^{(n)}) \right) \delta q^{(n)}\Delta t,
\end{aligned}$$

where D_1 and D_2 are derivatives with respect to the first and second variables, respectively. Note that the final equality uses $\delta q^{(0)} = \delta q^{(N)} = 0$. To be zero for any variation $\delta q^{(n)}$, the following conditions must be satisfied

$$D_1\mathcal{L}_\mathrm{d}(q^{(n)}, q^{(n+1)}) + D_2\mathcal{L}_\mathrm{d}(q^{(n-1)}, q^{(n)}) = 0.$$

This is an approximation to the Euler–Lagrange equation and is known as the discrete Euler–Lagrange equation.

5.2 Variational Integrator for Hamiltonian Neural Networks

5.2.1 Variational principle and variational integrator for Hamiltonian neural networks

Similar to the Euler–Lagrange equation, the Hamilton equation is known to be derived by the variational principle. Instead of S, consider the following integral:

$$\int_0^T (p \cdot \dot{q} - H(q,p))\,\mathrm{d}t \tag{5.3}$$

By ignoring the higher-order terms and fixing both ends, the variation of this integral becomes

$$\begin{aligned}
&\int_0^T ((p+\delta p)\cdot(\dot{q}+\delta\dot{q}) - H(q+\delta q, p+\delta p))\,\mathrm{d}t - \int_0^T (p\cdot\dot{q} - H(q,p))\,\mathrm{d}t \\
&= \int_0^T (p\cdot\delta\dot{q} + \delta p\cdot\dot{q} - D_1 H\delta q - D_2 H\delta p)\,\mathrm{d}t \\
&= \int_0^T \left(-\dot{p}\cdot\delta q + \delta p\cdot\dot{q} + [p\cdot\delta q]_0^T - \frac{\partial H}{\partial q}\cdot\delta q - \frac{\partial H}{\partial p}\cdot\delta p\right)\mathrm{d}t \\
&= \int_0^T \left((-\dot{p} - \frac{\partial H}{\partial q})\cdot\delta q + (\dot{q} - \frac{\partial H}{\partial p})\cdot\delta p\right)\mathrm{d}t
\end{aligned}$$

For this variation to be zero for any δq and δp, the following conditions must be hold:

$$-\dot{p} - \frac{\partial H}{\partial q} = 0, \quad \dot{q} - \frac{\partial H}{\partial p} = 0.$$

This corresponds to the Hamilton equation (5.1).

Next, we propose a variational integrator for Hamiltonian neural networks. Suppose that a trained Hamiltonian neural network is given, of which Hamiltonian is given by a neural network H_{NN}. First, as in the Lagrangian formalism, consider the following sum that approximates the integral (5.3)

$$\sum_{n=0}^{N-1} \left(p^{(n)}\cdot\frac{q^{(n+1)} - q^{(n)}}{\Delta t} - H_{\mathrm{NN}}(q^{(n)}, p^{(n)})\right)\Delta t$$

where $q^{(n)}$ and $p^{(n)}$ are approximations of $q(n\Delta t)$ and $p(n\Delta t)$, respectively. Computing the variation of the above sum with respect to the infinitesimal perturbations

$\delta q^{(n)}$ and $\delta p^{(n)}$ of $q^{(n)}$ and $p^{(n)}$ under the assumption that $\delta q^{(0)} = \delta q^{(N)} = 0$, we obtain

$$\begin{aligned}
&\sum_{n=0}^{N-1}\Big(p^{(n)} + \delta p^{(n)} \cdot \big(\frac{q^{(n+1)} - q^{(n)}}{\Delta t} + \frac{\delta q^{(n+1)} - \delta q^{(n)}}{\Delta t}\big) \\
&- H_{\mathrm{NN}}(q^{(n)} + \delta q^{(n)}, p^{(n)} + \delta p^{(n)})\Big)\Delta t \\
&- \sum_{n=0}^{N-1}\left(p^{(n)} \cdot \frac{q^{(n+1)} - q^{(n)}}{\Delta t} - H_{\mathrm{NN}}(q^{(n)}, p^{(n)})\right)\Delta t \\
&= \sum_{n=0}^{N-1}\Big(p^{(n)} \cdot \frac{\delta q^{(n+1)} - \delta q^{(n)}}{\Delta t} + \delta p^{(n)} \cdot \frac{q^{(n+1)} - q^{(n)}}{\Delta t} \\
&- D_1 H_{\mathrm{NN}}(q^{(n)}, p^{(n)}) \cdot \delta q^{(n)} - D_2 H_{\mathrm{NN}}(q^{(n)}, p^{(n)}) \cdot \delta p^{(n)}\Big)\Delta t.
\end{aligned}$$

For the first term, the following equality holds:

$$\begin{aligned}
\sum_{n=0}^{N-1} p^{(n)} \cdot \frac{\delta q^{(n+1)} - \delta q^{(n)}}{\Delta t}\Delta t &= \sum_{n=0}^{N-1} p^{(n)} \cdot \delta q^{(n+1)} - \sum_{n=0}^{N-1} p^{(n)} \cdot \delta q^{(n)} \\
&= \sum_{n=1}^{N} p^{(n-1)} \cdot \delta q^{(n)} - \sum_{n=0}^{N-1} p^{(n)} \cdot \delta q^{(n)} \\
&= \sum_{n=0}^{N-1} p^{(n-1)} \cdot \delta q^{(n)} - \sum_{n=0}^{N-1} p^{(n)} \cdot \delta q^{(n)} \\
&= \sum_{n=0}^{N-1}\left(-\frac{p^{(n)} - p^{(n-1)}}{\Delta t} \cdot \delta q^{(n)}\right)\Delta t
\end{aligned}$$

where $\delta q^{(0)} = \delta q^{(N)} = 0$ is used. This is often referred to as "summation by parts." Using this, the above variation can be rewritten as follows:

$$\begin{aligned}
&\sum_{n=0}^{N-1}\Big(-\frac{p^{(n)} - p^{(n-1)}}{\Delta t} \cdot \delta q^{(n)} + \delta p^{(n)} \cdot \frac{q^{(n+1)} - q^{(n)}}{\Delta t} \\
&- D_1 H_{\mathrm{NN}}(q^{(n)}, p^{(n)})\delta q^{(n)} - D_2 H_{\mathrm{NN}}(q^{(n)}, p^{(n)})\delta p^{(n)}\Big)\Delta t \\
&= \sum_{n=0}^{N-1}\Bigg(\Big(-\frac{p^{(n)} - p^{(n-1)}}{\Delta t} - D_1 H_{\mathrm{NN}}(q^{(n)}, p^{(n)})\Big) \cdot \delta q^{(n)} \\
&+ \Big(\frac{q^{(n+1)} - q^{(n)}}{\Delta t} - D_2 H_{\mathrm{NN}}(q^{(n)}, p^{(n)})\Big) \cdot \delta p^{(n)}\Bigg)\Delta t.
\end{aligned}$$

Hence for the vatiation to be zero the following equation must hold:

$$
\begin{aligned}
&-\frac{p^{(n)}-p^{(n-1)}}{\Delta t}-D_1H_{\mathrm{NN}}(q^{(n)},p^{(n)})=0,\\
&\frac{q^{(n+1)}-q^{(n)}}{\Delta t}-D_2H_{\mathrm{NN}}(q^{(n)},p^{(n)})=0.
\end{aligned}
$$

These equations can be rearranged to

$$
\begin{pmatrix}
\dfrac{q^{(n+1)}-q^{(n)}}{\Delta t}\\[2ex]
\dfrac{p^{(n)}-p^{(n-1)}}{\Delta t}
\end{pmatrix}
=\begin{pmatrix} O & I\\ -I & O\end{pmatrix}
\begin{pmatrix} D_1H_{\mathrm{NN}}(q^{(n)},p^{(n)})\\ D_2H_{\mathrm{NN}}(q^{(n)},p^{(n)})\end{pmatrix}. \tag{5.4}
$$

It turns out that this is certainly an approximation of the Hamilton equation (5.1).

We now consider whether $q^{(n+1)}$ and $p^{(n+1)}$ can be computed using this equation for a Hamiltonian H_{NN} given by a neural network. First, because each expression of (5.4) is valid for any n, so (5.4) can be rewritten as

$$
\begin{pmatrix}
\dfrac{q^{(n+1)}-q^{(n)}}{\Delta t}\\[2ex]
\dfrac{p^{(n+1)}-p^{(n)}}{\Delta t}
\end{pmatrix}
=\begin{pmatrix} O & I\\ -I & O\end{pmatrix}
\begin{pmatrix} D_1H_{\mathrm{NN}}(q^{(n)},p^{(n)})\\ D_2H_{\mathrm{NN}}(q^{(n+1)},p^{(n+1)})\end{pmatrix}. \tag{5.5}
$$

This defines the simultaneous equations for $q^{(n+1)},p^{(n+1)}$, when $q^{(n)},p^{(n)}$ are given. This system of equations can be solved numerically as explained below. The D_1 and D_2 are the derivatives of the neural network from the first and second variables, respectively. Therefore, if the values of $H_{\mathrm{NN}}(q^{(n)},p^{(n)})$ and $H_{\mathrm{NN}}(q^{(n+1)},p^{(n+1)})$ can be calculated, D_1H_{NN} and D_2H_{NN} can be obtained by using automatic differentiation.

For simplicity, let us solve the above system of equations by using a simple fixed-point iteration method. Let $q_{(k)}^{(n+1)},p_{(k)}^{(n+1)}$ be the approximations of $q^{(n+1)},p^{(n+1)}$ at the kth iteration. The algorithm of the fixed-point iteration method is as follows:

$$
\begin{aligned}
&q_{(0)}^{(n+1)}=q^{(n)},p_{(0)}^{(n+1)}=p^{(n)},\\
&\begin{pmatrix} q_{(k+1)}^{(n+1)}\\ p_{(k+1)}^{(n+1)}\end{pmatrix}
=\begin{pmatrix} q^{(n)}\\ p^{(n)}\end{pmatrix}
+\Delta t\begin{pmatrix} O & I\\ -I & O\end{pmatrix}
\begin{pmatrix} D_1H_{\mathrm{NN}}(q^{(n)},p^{(n)})\\ D_2H_{\mathrm{NN}}(q_{(k)}^{(n+1)},p_{(k)}^{(n+1)})\end{pmatrix}.
\end{aligned}
$$

The right-hand side can be computed using automatic differentiation. Thus, when the algorithm converges, the numerical method (5.5) certainly determines $q^{(n+1)},p^{(n+1)}$.

5.2.2 Numerical example

We trained Hamiltonian neural networks using a data set of a simple harmonic oscillator:

$$\dot{q} = p, \quad \dot{p} = -q.$$

We performed physical simulations by using the trained model discretized by the explicit Euler method and the proposed variational integrator. The time step size was set to $\Delta t = 0.01$. The numerical solutions are computed from $t = 0$ to $t = 100$. When the neural network was trained, it was confirmed that the loss function was certainly small enough. Fig. 5.1 shows the simulation results by the two methods, while Fig. 5.2 shows the energy behaviors. When the explicit Euler method was employed, the energy increased and the numerical solution diverged. When the proposed method was used, however, the energy was very well conserved and the solution continued to oscillate within a certain range.

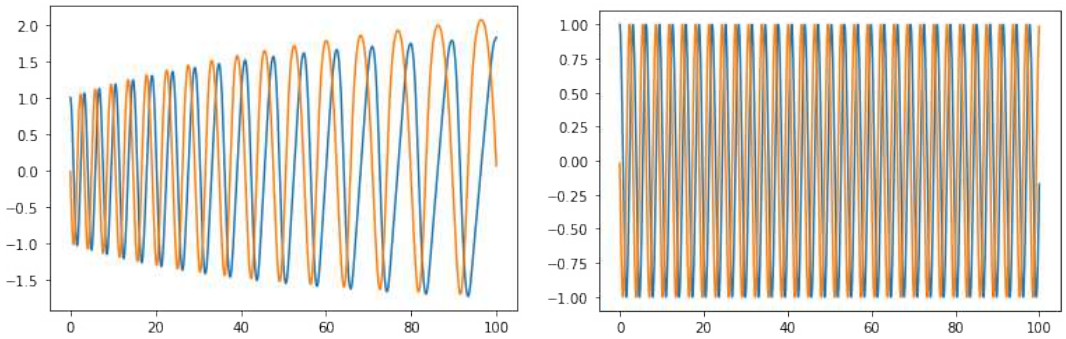


Fig.5.1 Predicted solutions (q: blue, p: orange) by the Euler method (left) and those by the variational integrator (right).

5.3 Variational Integrator for Neural Symplectic Forms

5.3.1 Variational principle and variational integrator for neural symplectic forms

As is well-known the Hamilton equation (5.1) in the canonical form is derived from the variational principle using the action integral (5.3) [1]. We need a similar action integral that is compatible with the neural symplectic form. The following is the

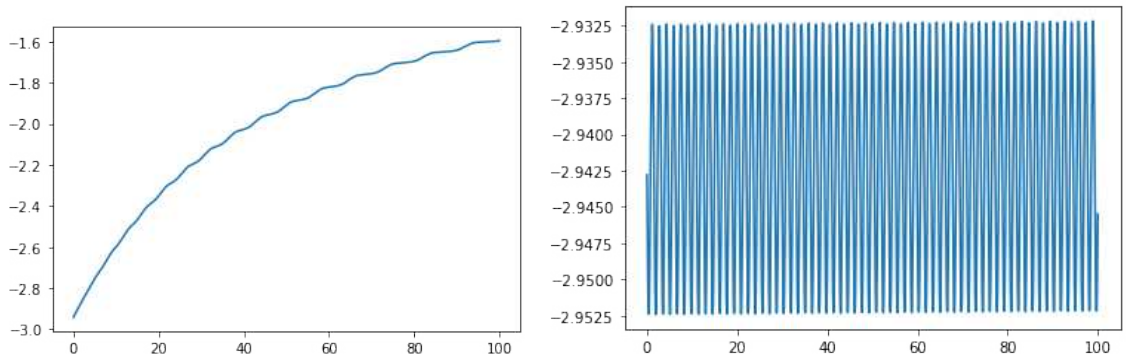


Fig.5.2 Predicted energies by the Euler method (left) and by the variational integrator (right).

model of neural symplectic form on the configuration manifold $\mathcal{M}$:

$$\tilde{\omega} = \mathrm{d}\theta_{\mathrm{NN}}, \quad \frac{\mathrm{d}u}{\mathrm{d}t} = \tilde{X}_{H_{\mathrm{NN}}}, \quad \omega(\tilde{X}_{H_{\mathrm{NN}}}, v) = \mathrm{d}H_{\mathrm{NN}}(v) \text{ for all } v \in T_u\mathcal{M}. \tag{5.6}$$

H_{NN} is the Hamiltonian and θ_{NN} is the symplectic 1-form, which are modeled by using multi-layer perceptrons. $\tilde{\omega}$ is the symplectic 2-form. To be a symplectic form $\tilde{\omega}$ must be closed; $\mathrm{d}\tilde{\omega} = 0$. To ensure this property, $\tilde{\omega}$ is modeled as $\tilde{\omega} = \mathrm{d}\theta_{\mathrm{NN}}$. Because of the property of the exterior derivative $\mathrm{dd} = 0$, this ensures the learned 2-form is closed; $\mathrm{d}\tilde{\omega} = \mathrm{dd}\theta_{\mathrm{NN}} = 0$. X_{NN} is a vector field that is defined by the last equation of (5.6). See [16] or Chapter 3 for details.

We focus on the first term $p\mathrm{d}q$ of the action integral (5.3). Taking the exterior derivative of it gives the symplectic form: $\mathrm{d}(p\mathrm{d}q) = -\mathrm{d}q \wedge \mathrm{d}p$, which is similar to $-\theta_{\mathrm{NN}}$ in neural symplectic form. Therefore, instead of (5.3), we should consider the following integral:

$$\int_0^T (-\theta_{\mathrm{NN}}(\dot{u}) - H_{\mathrm{NN}})\,\mathrm{d}t, \tag{5.7}$$

In fact, the variational principle associated with this integral derives neural symplectic form. For details, we consider the variation of the action integral (5.7) of neural symplectic form with infinitesimal perturbations Δu with respect to u with the both ends fixed $\Delta u^{(0)} = \Delta u^{(T)} = 0$:

$$\int_0^T (-\theta_{\mathrm{NN}}(u + \Delta u) \cdot (\dot{u} + \Delta\dot{u}) - H_{\mathrm{NN}}(u + \Delta u))\,\mathrm{d}t - \int_0^T (-\theta_{\mathrm{NN}}(u) \cdot (\dot{u}) - H_{\mathrm{NN}}(u))\,\mathrm{d}t.$$

Using the Taylor expansion to the above equation, we get

$$\int_0^T \left(-\theta_{\mathrm{NN}}(u) \cdot \Delta \dot{u} - \frac{\partial \theta_{\mathrm{NN}}}{\partial u} \Delta u \cdot \dot{u} - \frac{\partial H_{\mathrm{NN}}}{\partial u} \Delta u \right) \mathrm{d}t,$$

where higher-order terms of Δu are omitted. Next, by the integration by parts we obtain

$$\int_0^T \left(\frac{\mathrm{d}}{\mathrm{d}t} \theta_{\mathrm{NN}}(u) \cdot \Delta u - \dot{u}^T \frac{\partial \theta_{\mathrm{NN}}}{\partial u} \Delta u - \frac{\partial H_{\mathrm{NN}}}{\partial u} \Delta u \right) \mathrm{d}t.$$

Then, from the chain rule, we have

$$\int_0^T \left[\left(\dot{u}^T \left(\frac{\partial \theta_{\mathrm{NN}}}{\partial u}^T - \frac{\partial \theta_{\mathrm{NN}}}{\partial u} \right) - \frac{\partial H_{\mathrm{NN}}}{\partial u} \right) \Delta u \right] \mathrm{d}t.$$

For this variation to be zero for any Δu, the following equation must be satisfied

$$\dot{u}^T \left(\frac{\partial \theta_{\mathrm{NN}}}{\partial u}^T - \frac{\partial \theta_{\mathrm{NN}}}{\partial u} \right) - \frac{\partial H_{\mathrm{NN}}}{\partial u} = 0 \quad \Longleftrightarrow \quad \left(\frac{\partial \theta_{\mathrm{NN}}}{\partial u}^T - \frac{\partial \theta_{\mathrm{NN}}}{\partial u} \right)^T \dot{u} = \frac{\partial H_{\mathrm{NN}}}{\partial u}. \tag{5.8}$$

As show in [16] or Chapter 3, the neural symplectic form is known to be represented by in terms of vectors and a matrix, without using the differential forms:

$$\dot{u} = \tilde{W}_u^{-\top} \nabla H_{\mathrm{NN}}(u), \quad (\tilde{W}_u)_{i,j} = \frac{\partial (\theta_{\mathrm{NN}})_i}{\partial u_j} - \frac{\partial (\theta_{\mathrm{NN}})_j}{\partial u_i},$$

which is corresponded by the derived equality (5.8).

Note that all quantities that are needed to define the above integral are available in the neural symplectic form model (5.6).

The above action integral can be used to derive variational integrators. Although any discretization can be employed, we discretize the above action integral (5.7), for example, as

$$\sum_{n=0}^{N} \left(-\theta_{\mathrm{NN}}\left(\frac{u^{(n+1)} - u^{(n)}}{\Delta t}; \frac{u^{(n+1)} + u^{(n)}}{2} \right) - H_{\mathrm{NN}}\left(\frac{u^{(n+1)} + u^{(n)}}{2} \right) \right) \Delta t, \tag{5.9}$$

where Δt is the time step size and $u^{(n)}$ is an approximation of $u(n\Delta t)$. As for $\theta_{\mathrm{NN}}(\frac{u^{(n+1)}-u^{(n)}}{\Delta t}; \frac{u^{(n+1)}+u^{(n)}}{2})$, because θ_{NN} is a 1-form and a 1-form is a covector field, θ_{NN} defines a linear function $\theta_{\mathrm{NN}}(\cdot; \frac{u^{(n+1)}+u^{(n)}}{2}) \in T_{\frac{u^{(n+1)}+u^{(n)}}{2}} \mathcal{M}$ at each point $\frac{u^{(n+1)}+u^{(n)}}{2} \in \mathcal{M}$.

Computing the variation of discretized action integral (5.9) with respect to the infinitesimal perturbations $\Delta u^{(n)}$ of $u^{(n)}$ under the assumption that $\Delta u^{(0)} = \Delta u^{(N)} = 0$, we obtain

$$\sum_{n=0}^{N} \Bigg(-\theta_{\mathrm{NN}}\Big(\frac{u^{(n+1)} + \Delta u^{(n+1)} - u^{(n)} - \Delta u^{(n)}}{\Delta t}; \frac{u^{(n+1)} + \Delta u^{(n+1)} + u^{(n)} + \Delta u^{(n)}}{2}\Big) \\ -H_{\mathrm{NN}}\Big(\frac{u^{(n+1)} + \Delta u^{(n+1)} + u^{(n)} + \Delta u^{(n)}}{2}\Big) \Bigg) \Delta t - \sum_{n=0}^{N} \Bigg(-\theta_{\mathrm{NN}}\Big(\frac{u^{(n+1)} - u^{(n)}}{\Delta t}; \frac{u^{(n+1)} + u^{(n)}}{2}\Big) \\ -H_{\mathrm{NN}}\Big(\frac{u^{(n+1)} + u^{(n)}}{2}\Big) \Bigg) \Delta t,$$

Using the Taylor expansion to symplectic 1-form term θ_{NN} and Hamiltonian term H_{NN}, respectively, We get

$$\sum_{n=0}^{N} \Bigg(-D_1\theta_{\mathrm{NN}}\Big(\frac{u^{(n+1)} - u^{(n)}}{\Delta t}; \frac{u^{(n+1)} + u^{(n)}}{2}\Big)\frac{\Delta u^{(n+1)} - \Delta u^{(n)}}{\Delta t} \\ -D_2\theta_{\mathrm{NN}}\Big(\frac{u^{(n+1)} - u^{(n)}}{\Delta t}; \frac{u^{(n+1)} + u^{(n)}}{2}\Big)\frac{\Delta u^{(n+1)} + \Delta u^{(n)}}{2} \\ -DH_{\mathrm{NN}}\Big(\frac{u^{(n+1)} + u^{(n)}}{2}\Big)\frac{\Delta u^{(n+1)} + \Delta u^{(n)}}{2} \Bigg) \Delta t,$$

where DH_{NN} is the derivative of H_{NN}; $D_1\theta_{\mathrm{NN}}$ and $D_2\theta_{\mathrm{NN}}$ are derivatives with respect to the first and second variables of the θ_{NN}, respectively. The higher-order terms of $\Delta u^{(n)}$'s are omitted. Since θ_{NN} is a 1-form, θ_{NN} defines a linear map that maps a vector v to a real number $\theta_{\mathrm{NN}}(v; u)$ for each u. Because this is a linear map, there exists a vector $\vec{\theta}_{\mathrm{NN}}(u)$ such that $\theta_{\mathrm{NN}}(v; u) = \vec{\theta}_{\mathrm{NN}}(u) \cdot v$. Note that although $\vec{\theta}$ is a linear map with respect to the vector v, $\vec{\theta}$ can be nonlinearly dependent on u. By using this expression, $D_1\theta_{\mathrm{NN}}$ and $D_2\theta_{\mathrm{NN}}$ are given as

$$D_1\theta_{\mathrm{NN}}(v; u) = \frac{\partial}{\partial v}\vec{\theta}_{\mathrm{NN}}(u) = \vec{\theta}_{\mathrm{NN}}(u)$$
$$D_2\theta_{\mathrm{NN}}(v; u) = \frac{\partial}{\partial u}\vec{\theta}_{\mathrm{NN}}(u) \cdot v = J^T v,$$

where J is the Jacobian matrix: $J = \frac{\partial \vec{\theta}_{\mathrm{NN}}}{\partial u}$.

Rearranging the above equation, we get

$$\sum_{n=0}^{N}\Bigg[\Bigg(\frac{1}{\Delta t}D_1\theta_{\mathrm{NN}}(\frac{u^{(n+1)}-u^{(n)}}{\Delta t};\frac{u^{(n+1)}+u^{(n)}}{2})-\frac{1}{2}D_2\theta_{\mathrm{NN}}(\frac{u^{(n+1)}-u^{(n)}}{\Delta t};\frac{u^{(n+1)}+u^{(n)}}{2})$$
$$-\frac{1}{2}DH_{\mathrm{NN}}(\frac{u^{(n+1)}+u^{(n)}}{2})\Bigg)\Delta u^{(n)}\Bigg]\Delta t+\sum_{n=1}^{N+1}\Bigg[\Bigg(-\frac{1}{\Delta t}D_1\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})$$
$$-\frac{1}{2}D_2\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})-\frac{1}{2}DH_{\mathrm{NN}}(\frac{u^{(n)}+u^{(n-1)}}{2})\Bigg)\Delta u^{(n)}\Bigg]\Delta t.$$

The second term can be rewritten by using the assumption that the both ends are fixed, $\Delta u^{(0)}=\Delta u^{(N)}=0$, as:

$$\sum_{n=1}^{N+1}\Bigg[\Bigg(-\frac{1}{\Delta t}D_1\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})-\frac{1}{2}D_2\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})$$
$$-\frac{1}{2}DH_{\mathrm{NN}}(\frac{u^{(n)}+u^{(n-1)}}{2})\Bigg)\Delta u^{(n)}\Bigg]\Delta t$$
$$=\sum_{n=0}^{N}\Bigg[\Bigg(-\frac{1}{\Delta t}D_1\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})-\frac{1}{2}D_2\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})$$
$$-\frac{1}{2}DH_{\mathrm{NN}}(\frac{u^{(n)}+u^{(n-1)}}{2})\Bigg)\Delta u^{(n)}\Bigg]\Delta t.$$

We thus obtain

$$\sum_{n=0}^{N}\Bigg\{\Bigg[\frac{1}{\Delta t}\Bigg(D_1\theta_{\mathrm{NN}}(\frac{u^{(n+1)}-u^{(n)}}{\Delta t};\frac{u^{(n+1)}+u^{(n)}}{2})-D_1\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})\Bigg)$$
$$+\frac{1}{2}\Bigg(-D_2\theta_{\mathrm{NN}}(\frac{u^{(n+1)}-u^{(n)}}{\Delta t};\frac{u^{(n+1)}+u^{(n)}}{2})-DH_{\mathrm{NN}}(\frac{u^{(n+1)}+u^{(n)}}{2})$$
$$-D_2\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})-DH_{\mathrm{NN}}(\frac{u^{(n)}+u^{(n-1)}}{2})\Bigg)\Bigg]\Delta u^{(n)}\Bigg\}\Delta t.$$

Considering variations with respect to any $\Delta u^{(n))}$ to be zero, we require the following equality:

$$\frac{1}{\Delta t}\Bigg(D_1\theta_{\mathrm{NN}}(\frac{u^{(n+1)}-u^{(n)}}{\Delta t};\frac{u^{(n+1)}+u^{(n)}}{2})-D_1\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})\Bigg)$$
$$+\frac{1}{2}\Bigg(-D_2\theta_{\mathrm{NN}}(\frac{u^{(n+1)}-u^{(n)}}{\Delta t};\frac{u^{(n+1)}+u^{(n)}}{2})-DH_{\mathrm{NN}}(\frac{u^{(n+1)}+u^{(n)}}{2})$$
$$-D_2\theta_{\mathrm{NN}}(\frac{u^{(n)}-u^{(n-1)}}{\Delta t};\frac{u^{(n)}+u^{(n-1)}}{2})-DH_{\mathrm{NN}}(\frac{u^{(n)}+u^{(n-1)}}{2})\Bigg)=0,$$

which is the proposed variational integrator.

5.3.2 Numerical example

We illustrate the effectiveness and advantage of the proposed method in the following numerical experiment. Firstly, we trained the neural symplectic form on a Hamiltonian system, a double pendulum, using the code and data published in [16][*1]. In this experiment, the equation of the double pendulum used in the experiment is as follows

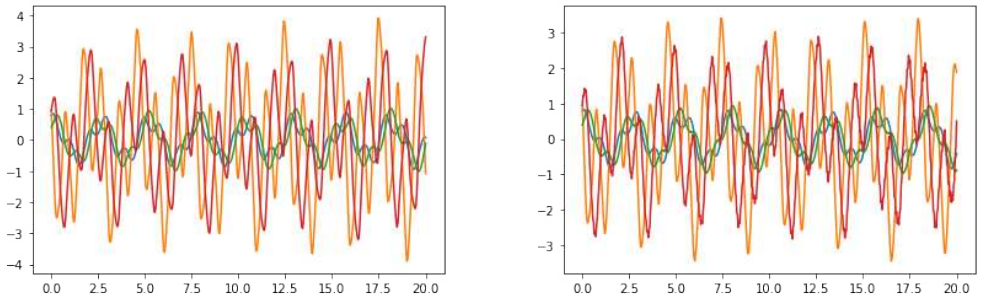
$$\begin{aligned}
&\frac{\mathrm{d}\theta_1}{\mathrm{d}t} = \phi_1, \quad \frac{\mathrm{d}\theta_2}{\mathrm{d}t} = \phi_2, \\
&\frac{\mathrm{d}\phi_1}{\mathrm{d}t} = \frac{g(\sin\theta_2 \sin(\theta_1-\theta_2) - \frac{m_1+m_2}{m_2}\sin(\theta_1)) - (l_1\theta_1^2\cos(\theta_1-\theta_2) + l_2\theta_2^2)\sin(\theta_1-\theta_2)}{l_1(\frac{m_1+m_2}{m_2} - \cos^2(\theta_1-\theta_2))}, \\
&\frac{\mathrm{d}\phi_2}{\mathrm{d}t} = \frac{\frac{g(m_1+m_2)}{m_2}(\sin\theta_1\cos(\theta_1-\theta_2) - \sin(\theta_2)) - (\frac{l_1(m_1+m_2)}{m_2}\theta_1^2 + l_2\theta_2^2\cos(\theta_1-\theta_2))\sin(\theta_1-\theta_2)}{l_2(\frac{m_1+m_2}{m_2} - \cos^2(\theta_1-\theta_2))}.
\end{aligned}$$

We also explained why it is difficult to learn this equation with a Hamiltonian neural network and why a neural symplectic form is necessary in Chapter 3.

The training conditions are the same as in [16] or Chapter 3. The Hamiltonian H_{NN} and the 1-form θ_{NN} were modeled by using a neural network with two hidden layers of 200 units and the tanh activation function. We trained the model using the Adam optimizer with a learning rate of 10^{-3} for 2000 iterations. All computations are performed by using NVIDIA A100. The time step Δt was set to 0.04.

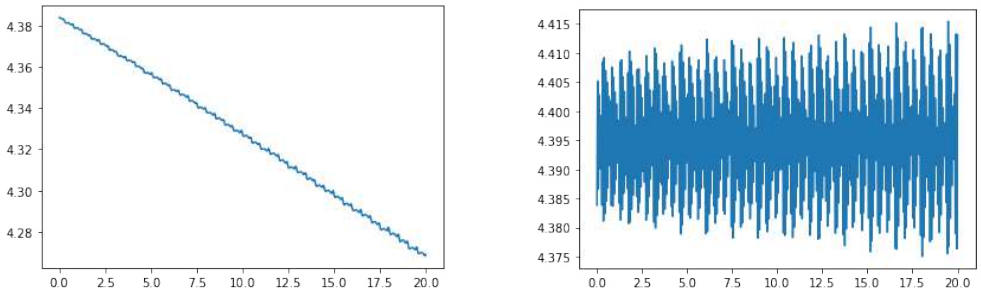
Next, the model was discretised using the proposed method. The proposed method is symmetric and hence has a second-order accuracy [40]. Therefore, we compared it with the Heun method, which also has a second-order accuracy. The calculated trajectories and the energies learned by the neural network are shown in Fig. 5.3 and Fig. 5.4, respectively. Although the trajectories are not significantly different, the energy graphs show that the energy by the Heun method gradually increases, while that by the proposed method oscillates, neither diverging nor decaying. This confirms the energy conservation property of the proposed method.

[*1] `https://github.com/YuhanChen0805/neural_symplectic_form` (MIT License)



(a) The Heun method (b) Proposed variational integrator

Fig.5.3 Example of the orbits simulated by the numerical methods. Each component of $u(t) = (\theta_1(t), \phi_1(t), \theta_2(t), \phi_2(t))$ is represented: blue (θ_1), orange (ϕ_1), green (θ_2), and red (ϕ_2).The horizontal axis represents the time and the vertical axis represents the values of the variables.



(a) The Heun method (b) Proposed variational integrator

Fig.5.4 Example of the state of the conservation of the energies predicted by the proposed and the comparative numerical method. The horizontal axis represents time and the vertical axis represents energy.

Chapter 6 Super Resolution of Numerical Solutions of Nonlinear Elliptic Equations by DeepONet

In this chapter, we propose to employ DeepONet's network architecture for super resolution of numerical solutions of partial differential equations. Because the output of the DeepONet is a function, our method can achieve super resolution of the results of physical simulations regardless of the scale factors or the discretization method.

This chapter is organized as follows. First, in Section 6.1, we show a biref introduction to super resolution and neural operators. Section 6.2 explains the nonlinear elliptic equation and a numerical method for this equation. Next, in Section 6.3, we explain the proposed super resolution method with DeepONet. Finally, in Section 6.4, we show the results of numerical experiments, which illustrate the validity of our proposed method.

6.1 Related Work with Super Resolution and Neural Operators

Super resolution amplifies the resolution of images to obtain clearer images. This method is used for general image processing and ultra-high resolution microscopy by reconstructing high-resolution images from low-resolution counterparts. Meanwhile, deep neural networks have been widely applied to solving partial differential equations in a data-driven manner without using numerical solvers [49, 80, 124, 127]. In particular, super-resolution modeling of sparse observations is used in the field of physics [96]. However, most deep neural networks for super resolution are developed in configurations with a single scale factor, which cannot be used in scenarios re-

quiring arbitrary super-resolution factors [118, 129]. Once trained on a specific grid size, the neural network is limited to that resolution. In the field of physics, solving partial differential equations at different resolution is often necessary to investigate the systems in more detail.

In recent years, neural operators have been proposed as deep neural networks for learning mapping relations between infinite-dimensional function spaces, and they are actively used in obtaining numerical solutions for partial differential equations [57]. Neural operators can approximate any given nonlinear continuous operator in a data-driven manner and perform well in many applications. Neural operators typically use the kernel integrals on the spatial domain, thereby explicitly capturing the global relations that are needed for learning the underlying solution operators of the partial differential equations. While conventional numerical solvers of partial differential equations take a long time to simulate the dynamics, neural operators can yield approximate solutions much faster. For example, DeepONet was proposed for approximating operators based on a universal approximation theorem [61, 71]. Other methods such as Fourier Neural Operator [67], Spectral Neural Operators [28], and Super-Resolution Neural Operator [120] have also been proposed. Training these models require input functions and solutions as the training data for each partial differential equation. Typically, these data are generated using numerical solvers. The network weights are iteratively adjusted until the entire network approximates the target operator as closely as possible.

6.2 Target Nonlinear Elliptic Equations

The nonlinear elliptic equation is one of the types of partial differential equations describing phenomena that do not change from moment to moment. In real physical phenomena such as weather systems or thermodynamic systems the change in the independent variable of a given system is mostly not proportional to the change in the output. Therefore, our study discusses the case of nonlinear elliptic equations. More precisely, we consider the following nonlinear Poisson equation on the interval $[0, a]$ under the Dirichlet boundary condition as a typical example:

$$u_{xx} = f(u, x), \quad u = u(x), \quad u(0) = u(a) = 0 \tag{6.1}$$

where u_{xx} is the second order partial derivative of u with respect to x [89].

In this study, we suppose that the standard finite difference method is used to discretize the equation. The finite-difference method is a discretization approach to obtain the numerical solution of the equation. This method defines a grid in the region where the solution is to be obtained, and the partial differentials are approximated by the finite differences between the grid points, thereby reducing the partial differential equations to difference equations [35].

Suppose that the grid has $M + 1$ nodes so that the region is divided into M parts with the step size $h_x = a/M$.

The positions of the nodes are denoted by x_i and the approximated solution at each x_i is denoted by u_i. By discretizing the differential operators by the central difference method(6.1), we obtain

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h_x^2} = f(u_i) \quad (i = 2, ..., M), \quad u_1 = u_{M+1} = 0 \tag{6.2}$$

An approximate solution is obtained by solving the above difference equations (6.2). However, the step size h_x must be sufficiently small to yield highly accurate numerical solutions necessary for physics, which is often computationally expensive.

6.3 Super Resolution with DeepONet

In conventional deep learning problems, approximating functions is the typical problem setting. On the other hand, neural operators can learn mappings between function spaces [57]. This implies neural operators can be applied to the super resolution; we propose such a method in this study.

In our proposed method, we employ DeepONet for learning the mapping between low-resolution numerical solutions and high-resolution ones so that the trained model can perform super resolution of physical simulations. In particular, because the output of DeepONet is a function we can obtain the super-resolution results at arbitrary scales by the proposed method.

The details of the proposed method are as follows. Let $\mathcal{B}(Q)$ be a certain Banach space of functions of the solutions to the target equation. For example, for the nonlinear Poisson equation, Q is the interval $[0, a]$ and $\mathcal{B}(Q)$ is the Sobolev space H^1 since

the nonlinear Poisson equation can be formulated as the weak form:

$$-\int_0^a u_x v_x \mathrm{d}x = \int_0^a f(u,x)v(x)\mathrm{d}x \qquad \text{for all } v,$$

which requires only the 1st order differentiability. Because the low-resolution numerical solutions are finite dimensional vectors in $\mathbb{R}^{M+1}$, we want to approximate the operator $G : \mathbb{R}^{M+1} \to \mathcal{B}(Q) = H^1([0,a])$; we apply DeepONet to this task.

DeepONet defines an operator G_θ between two Banach spaces with the trainable parameters θ. DeepONet has a bifurcated structure, processing input in two parallel networks: the branch and the trunk. Each of these networks are defined as a standard dense neural network. The "branch" network takes a finite number of sampled values of the input function; in the proposed method we input the whole low-resolution numerical solutions of the target partial differential equation without sampling since the numerical solutions are finite dimensional. The "trunk" network takes the spatial coordinate $x \in Q$ as the input. In our case, we input the position $x \in [0,a]$ at which the high-resolution solution should be evaluated.

In order to train the proposed model, we should minimize the following empirical-risk

$$\min_\theta \frac{1}{N}\sum_{k=1}^{N}\int_0^a |u^{(k)} - G_\theta(a^{(k)})|^2 \mathrm{d}x, \tag{6.3}$$

where $\{(a^{(k)}, u^{(k)}) \mid a^{(k)} \in \mathbb{R}^{M+1}, u^{(k)} \in H^1([0,a]), k = 1,\dots,N\}$ is the training data set. For the minimization problem (6.3) to be tractable, the L^2 norm should be discretized by a numerical integration method, which requires a finite numbers of sample points of the integrand. Hence, what we actually need as the target data is a set of a finite number of observations of the target solutions $\{u^{(k)}(\hat{x}_1), \dots, u^{(k)}(\hat{x}_{\hat{M}+1}$ |k=1,..., N)$\}$.

In the proposed method, to generate these data, we solve the difference equation with a coarse step size h_s for low-resolution numerical solutions for the input data and with a fine step size h_d for high-resolution numerical solutions used to evaluate the discretized empirical-risk.

6.4 Numerical Example

To test the proposed method, we apply our method to the following nonlinear Poisson equation

$$u_{xx} = u^2 + f_{\text{rand}}(x) \tag{6.4}$$

$f_{\text{rand}}(x)$ is given as a randomly generated polynomial. The data used to train the model are obtained by solving the difference equations with Scipy fslove.

In this experiment we used the set of numerical solutions each of which is associated with a different $f_{\text{rand}}(x)$. The low-resolution data are computed with the sparse grid with 11 nodes, i.e., $M = 10$. The high-resolution data is the set of sampled values of the solutions to the weak form of the nonlinear Poisson equation. We employed the trapezoidal rule to discretize the L^2 norm in the loss function (6.3). We performed two experiments in which 200 and 1000 sample points are used to evaluate the loss function, respectively. The DeepOnet were modeled using two neural networks with the same hidden layer of 200 units and the tanh activation function. We used 80 percent of generated data for training and the remaining for the test. We trained the model with a learning rate of 10^{-4} for 10000 iterations.

After the neural networks were trained, it was confirmed that the loss function was certainly small enough. In this experiment, the value of the loss function under the 200 sample points and 1000 sample points decreases to 4.74×10^{-5} and 9.82×10^{-6}, respectively.

Examples of the output of super resolution by the proposed method are shown in Figure 6.1, while Figure 6.2 shows the graph of the corresponding function f_{rand} in (6.4). As expected, the output of our model fit the real solution very well.

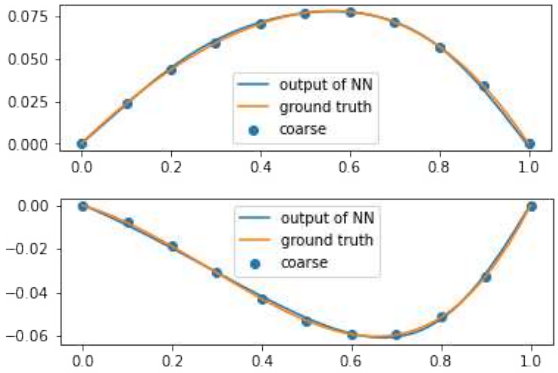


Fig.6.1 The learning results of the proposed method by the 200 sample points(top) and 1000 sample points(bottom). The scatters are the input low-resolution numerical solutions. The orange orbit is the ground truth, and the blue orbit is the output result of our proposed method.

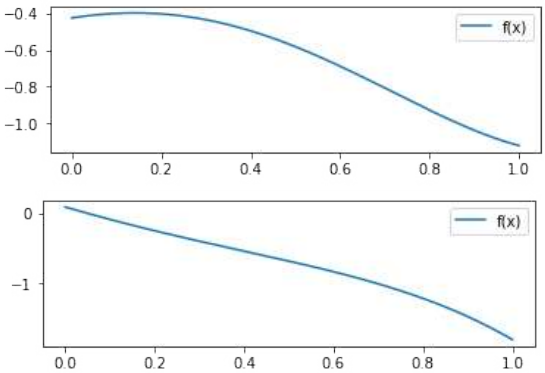


Fig.6.2 Polynomial functions with random parameters in the nonlinear Poisson equation with 200 sample points(top) and 1000 sample points(bottom). Note that these functions are not input into the neural network model.

Chapter 7 Conclusion

Artificial neural networks have been shown to have a universal approximation theorem, which states that the model has the ability to approximate any function. In the field of physics, most physical phenomena can be represented by equations of motion as differential equations. The study of neural network-based physical models is of great research value. In this thesis, we explore deep learning in physics. Specifically, we have proposed several novel deep physics models along with theoretical analyses and numerical experiments. In addition, structure-preserving numerical integrators for deep physics models and a method of super-resolution using a neural operator are also proposed. Thus, this thesis is a broad study regarding the potential of deep learning to contribute to physical modelling and simulation. Below, we provide a brief summary and discuss future work for each study.

Firstly, as an application of neural networks to physical modelling, we establish a more stable and secure communication system by improving the encryption system proposed in [101]. The summary and future work regarding this study is as follows.

- This study proposes a confidential communication system for color image communication using a chaotic synchronous distribution system and deep learning. We use a neural network to approximate the van der Pol boundary condition, so that the learned neural network can exhibit chaotic behavior as the original boundary condition does. In the construction of the neural network, we consider the middle layer of the hidden layer as an output layer as well. The number of neurons in the input and output layers is related to the input image, so that the network can be divided into two parts, corresponding to the left and right boundary conditions, respectively. After confirming that the proposed approach can be applied to simpler grayscale images, we further experimented with color images. The experimental results show that the neural network still

learns chaotic phenomena, and it can hide and restore the images well.

- Several security tests were also performed. The proposed method is not designed to encrypt the images as the pseudo-random sequences. Therefore, statistical tests were not appropriate, and in many cases AES produced images that were closer to random numbers. However, in terms of the UACI value, the proposed method was superior in some cases. On the other hand, if the encrypted images are identical regardless of the original image, then it becomes difficult to infer the original image from the encrypted image. We performed some tests from this perspective as well. The results imply that the encrypted images from the proposed method are almost the same regardless of the original images, thereby showing that the proposed method is certainly secure.
- Although we have confirmed to some extent that the techniques used in this study can be applied to color images, there is still room for improvement. In particular, the neural network in this study learns chaotic phenomena by approximating $qF_{\alpha,\beta}$, which is still somewhat risky. In the future, instead of approximating a function, new activation functions will be created to make the network behave in a chaotic manner.

Secondly, we have investigated deep physics models. Deep physics models are neural network models that can learn equations of motion representing physical phenomena. Therefore the most important feature is the ability to retain physical properties while approximating the equations. Such models include Hamiltonian neural networks and Lagrangian neural networks. Based on these existing deep physical models, we propose a new model that can learn the coordinate-free form of the Hamilton equation, which overcomes the shortcomings of existing models. This is one of the significant achievements of this thesis.

- We clarify the advantages and disadvantages of the existing models, such as (1) both models are proposed based on analytical mechanics and thus can retain physical properties well; (2) from the geometric mechanic, Hamiltonian neural networks have wider applications than Lagrangian neural networks since the Euler-Lagrange equation is only partially equivalent to Hamiltonian equations; (3) the generalized momentum in Hamiltonian neural networks is dependent on the unknown energy Hamiltonian H, which makes data preparation very

difficult; (4) it is possible that the learning outcome of skew matrix learning is not a Hamilton equation, which makes learning less efficient.

- We proposed a method for learning Hamilton equations from data represented on general coordinate systems, which are not restricted to generalized momenta. The key ingredient is the neural symplectic form; we proposed to learn the symplectic 2-form by using neural networks from data, thereby learning this coordinate-free representation. In particular, the Hamilton equation can be represented using a state-dependent skew-symmetric matrix, but not all skew-symmetric matrices are related to the symplectic 2-form. In the proposed method, in order to restrict the output of the model to the symplectic 2-forms, the 1-form that derives the symplectic 2-form is learned by the neural networks.
- Meanwhile, the proposed method requires the inverse of the skew-symmetric matrix, which may be computationally expensive for modeling large systems. To address this problem, the perturbation theory of the inverse matrix may be applied. For example, it is known that for a matrix M if the norm of ΔM is small enough, $(M + \Delta M)^{-1} \simeq M^{-1} + M^{-1}\Delta M M^{-1}$ holds. This should be investigated to reduce the computational costs in future work.

Thirdly, as stated above, deep physical models are mainly used for simulations; however, they must be discretized using numerical integrators for simulations, and unless carefully designed, numerical integrators destroy physical laws such as the energy conservation law. Numerical integrators that do not destroy physical laws are called structure-preserving integrators. Typical integrators are symplectic integrators, which can be derived as variational integrators. In this study, we show that variational integrators can be applied to both Hamiltonian neural networks and neural symplectic forms.

- For deep physical models, the variational calculus cannot be performed by hand because the energy function is given by a neural network. Therefore, it was necessary to confirm that this principle can be certainly applied and the solutions of the derived numerical method can be computed by using automatic differentiation. In addition, the energy conservation property of the proposed method was confirmed by the numerical experiment.
- The model equation in the neural symplectic form is not in the standard form

of the Hamilton equation that is usually assumed in the design of symplectic integrators, and hence the development of symplectic integrators is not straightforward. In this study, we focused on the fact that the action integral that derives the model of the neural symplectic forms can be expressed using the symplectic 1-form and Hamiltonian learned by the neural networks. Taking the variation of the discretized action integral expressed by the neural networks, we have proposed variational integrators for the neural symplectic forms. The numerical experiment has indeed confirmed the conservation of energy.

- Future work includes the development of more accurate numerical integrators by discretizing the action integral in a more sophisticated way.

In addition, super resolution amplifies the resolution of images to obtain clearer images. This method is used for general image processing and ultra-high resolution microscopy by reconstructing high-resolution images from low-resolution counterparts. Super resolution modelling of sparse observations of partial differential equations based on the conventional super-resolution techniques for images has a disadvantage that the deep neural network is only applicable to a single scale factor. Instead, neural operators that can learn the mapping relation between two infinite-dimensional function spaces. We present in this study a method capable of super resolution modelling at arbitrary scale factors by applying a neural operator, DeepONet.

- Deeponet, which is a type of neural opraters, can approximate the operator between function spaces. DeepONet has a characteristic architecture which processes data in two parallel networks, the branch and the trunk. DeepONet learns the solution operators for partial differential equations by taking the inner product of the outputs of these two networks. The approach we propose is to combine DeepONet with super resolution; in fact, since the output of DeepONet is a function, we can obtain the super-resolution results at arbitrary scales. To train this model, we solve the differential equation with a coarse step size for low-resolution numerical solutions for the input data and with a fine step size for high-resolution numerical solutions used to evaluate the discretized empirical risk.
- At present, our numerical experiments with super resolution of first-order partial differential equations have provided preliminary support that the proposed

approach is capable of generating high-resolution solutions at arbitrary scale factors.

- However, as the experimental subjects are relatively simple, numerical experiments on second-order nonlinear elliptic equations and even more complex partial differential equations are required.

Bibliography

[1] R. Abraham and J. E. Marsden. *Foundations of Mechanics*. American Mathematical Soc., 2008.

[2] R. A. Adams and J. J. F. Fournier. *Sobolev Spaces*. Elsevier, 2003.

[3] A. Alireza, R. M. Javad, and G. Behnam. An image encryption method based on chaos system and aes algorithm. *The Journal of Supercomputing*, 75:6663 – 6682, 2019.

[4] V. I. Arnol'd. *Mathematical Methods of Classical Mechanics*. Springer Science & Business Media, 2013.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv*, 2016.

[6] P. Bharadwaj, M. Li, and L. Demanet. SymAE: An autoencoder with embedded physical symmetries for passive time-lapse monitoring. In *SEG Technical Program Expanded Abstracts 2020*. Society of Exploration Geophysicists, 2020.

[7] R. Bondesan and A. Lamacraft. Learning Symmetries of Classical Integrable Systems. In *ICML 2019 Workshop on Theoretical Physics for Deep Learning*, 2019.

[8] O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In O. Bousquet, U. von Luxburg, and G. Rätsch, editors, *Advanced Lectures on Machine Learning*, pages 169–207. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[9] G. Caginalp. An analysis of a phase field model of a free boundary. *Archive for Rational Mechanics and Analysis*, 92(3):205–245, 1986.

[10] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis. Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(6), 2021.

[11] J. Caldeira, W. Wu, B. Nord, C. Avestruz, S. Trivedi, and K. Story. DeepCMB: Lensing reconstruction of the cosmic microwave background with deep neural networks. *Astronomy and Computing*, 28:100307, 2019.

[12] D. Chen, X. Gao, C. Xu, S. Wang, S. Chen, J. Fang, and Z. Wang. Flowdnn: a physics-informed deep neural network for fast and accurate flow prediction. *Frontiers Inf. Technol. Electron. Eng.*, 23(2):207–219, 2022.

[13] G. Chen, S.-B. Hsu, J. Zhou, G. Chen, and G. Crosta. Chaotic vibrations of the one-dimensional wave equation due to a self-excitation boundary condition part i: Controlled hysteresis. *Transactions of the American Mathematical Society*, 350(11):4265–4311, 1998.

[14] R. Chen and M. Tao. Data-driven prediction of general hamiltonian dynamics via learning exactly-symplectic maps. *Proceedings of the 38 th International Conference on Machine Learning, PMLR 139*, 2021.

[15] T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[16] Y. Chen, T. Matsubara, and T. Yaguchi. Neural symplectic form: Learning hamiltonian equations on general coordinate systems. In *Advances in Neural Information Processing Systems*, 2021.

[17] Y. Chen, T. Matsubara, and T. Yaguchi. Kam theory meets statistical learning theory: Hamiltonian neural networks with non-zero training loss, 2022.

[18] Y. Chen, H. Sano, M. Wakaiki, and T. Yaguchi. Secret communication systems using chaotic wave equations with neural network boundary conditions. *Entropy*, 23(7), 2021.

[19] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic Recurrent Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[20] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian Neural Networks. *ICLR 2020 Deep Differential Equations Workshop*, 2020.

[21] K. M. Cuomo and A. V. Oppenheim. Circuit implementation of synchronized chaos with applications to communications. *Phys. Rev. Lett.*, 71:65–68, 1993.

[22] S. Cuomo, V. S. di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific machine learning through physics-informed neural networks: Where we are and what's next, 2022.

[23] L. Deng. Deep learning: from speech recognition to language and multimodal processing. *APSIPA Transactions on Signal and Information Processing*, 5:e1, 2016.

[24] S. Desai and S. Roberts. VIGN: Variational Integrator Graph Networks. *arXiv*, 2020.

[25] D. M. DiPietro, S. Xiong, and B. Zhu. Sparse symplectically integrated neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[26] F. A. et al. Content based image retrieval (cbir) by statistical methods. *Baghdad Science Journal*, 17(2(SI)):0694, 2020.

[27] L. C. Evans. *Partial differential equations*. American Mathematical Society, Providence, R.I., 2010.

[28] V. Fanaskov and I. Oseledets. Spectral neural operators, 2022.

[29] S. Fatma, K. Sonia, Z. Medien, T. Rached, M. Mohsen, and B. Adel. High-level implementation of a chaotic and aes based crypto-system. *Journal of Circuits, Systems and Computers*, 26(07):1750122, 2017.

[30] Y. Feng, H. Wang, H. Yang, and F. Wang. Time-continuous energy-conservation neural network for structural dynamics analysis. *arXiv*, 2020.

[31] M. Finzi, S. Stanton, P. Izmailov, and A. G. Wilson. Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data. In *International Conference on Machine Learning (ICML)*, pages 3165–3176, 2020.

[32] M. Finzi, K. A. Wang, and A. G. Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[33] D. Furihata and T. Matsuo. *Discrete Variational Derivative Method: A Structure-Preserving Numerical Method for Partial Differential Equations*. Chapman and Hall/CRC, 2010.

[34] N. Galioto and A. A. Gorodetsky. Bayesian identification of hamiltonian dynamics from symplectic data. In *IEEE Conference on Decision and Control*

(CDC), pages 1190–1195, 2020.

[35] L. Gavete, F. Ureña, J. Benito, A. García, M. Ureña, and E. Salete. Solving second order non-linear elliptic partial differential equations using generalized finite difference method. *Journal of Computational and Applied Mathematics*, 318:378–387, 2017. Computational and Mathematical Methods in Science and Engineering CMMSE-2015.

[36] E. Giné and R. Nickl. *Mathematical Foundations of Infinite-Dimensional Statistical Models*. Cambridge University Press, 2016.

[37] C. Goong, H. Sze-Bi, and Z. Jianxin. Chaotic vibration of the wave equation with nonlinear feedback boundary control: Progress and open questions. *Chaos Control*, pages 25–50, 2004.

[38] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[39] K. Guan. Important notes on lyapunov exponents, 2014.

[40] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: Structure-preserving algorithms for ordinary differential equations*. Springer Series in Computational Mathematics. Springer, Berlin, Germany, 2002 edition, 2013.

[41] Z. He, F. Ni, W. Wang, and J. Zhang. A physics-informed deep learning method for solving direct and inverse heat conduction problems of materials. *Materials Today Communications*, 28:102719, 2021.

[42] B. Hernández-Bermejo and V. Fairén. Hamiltonian structure and darboux theorem for families of generalized Lotka–Volterra systems. *J. Math. Phys.*, 39(11):6162–6174, 1998.

[43] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990.

[44] X. Hu, H. Hu, S. Verma, and Z.-L. Zhang. Physics-guided deep neural networks for power flow analysis. *IEEE Transactions on Power Systems*, 36(3):2082–2092, 2021.

[45] N. Huang, M. Slaney, and M. Elhilali. Connecting deep neural networks to physical, perceptual, and electrophysiological auditory signals. *Frontiers in Neuroscience*, 12, 2018.

[46] Y. INOUE. Chaos. *JAPANESE JOURNAL OF MULTIPHASE FLOW*, 11(2):157–162, 1997.

[47] T. Iwaniec, John Raymond French Distinguished Professor of Mathematics Tadeusz Iwaniec, G. Martin, and of Mathematics Gaven Martin. *Geometric Function Theory and Non-linear Analysis*. Oxford University Press, 2001.

[48] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.

[49] C. Jiang, R. Vinuesa, R. Chen, J. Mi, S. Laima, and H. Li. An interpretable framework of data-driven turbulence modeling using deep neural networks. *Physics of Fluids*, 33(5), 2021.

[50] P. Jin, Z. Zhang, I. G. Kevrekidis, and G. E. Karniadakis. Learning poisson systems and trajectories of autonomous systems via poisson neural networks. *arXiv*, 2020.

[51] P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis. Sympnets: Intrinsic structure-preserving symplectic networks for identifying hamiltonian systems. *Neural Networks*, 132:166–179, 2020.

[52] Z. Jin, J. Y.-Y. Lin, and S.-F. Li. Learning principle of least action with reinforcement learning. *arXiv*, 2020.

[53] Y. K. Multichannel digital communications by the synchronization of globally coupled chaotic systems. *Physical Review E*, 60(2):1648–1657, 1999.

[54] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021.

[55] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[56] L. Kocarev and U. Parlitz. General approach for chaotic synchronization with applications to communication. *Phys. Rev. Lett.*, 74:5028–5031, 1995.

[57] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces, 2023.

[58] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and

K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[59] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing, 2016.

[60] C.-L. Kuo. Design of a fuzzy sliding-mode synchronization controller for two different chaos systems. *Computers and Mathematics with Applications*, 61(8):2090–2095, 2011. Advances in Nonlinear Dynamics.

[61] S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for deeponets: A deep learning framework in infinite dimensions, 2022.

[62] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[63] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[64] C. Li, F. Zhao, C. Liu, L. Lei, and J. Zhang. A hyperchaotic color image encryption algorithm and security analysis, 2019.

[65] L. Li, Y. Huang, and M. Xiao. Observer design for wave equations with van der pol type boundary conditions. *SIAM Journal on Control and Optimization*, 50(3):1200–1219, 2012.

[66] S.-H. Li, C.-X. Dong, L. Zhang, and L. Wang. Neural Canonical Transformation with Symplectic Flows. *Physical Review X*, 10(2):021020, 2020.

[67] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[68] C.-H. Lin, G.-H. Hu, C.-Y. Chan, and J.-J. Yan. Chaos-based synchronized dynamic keys and their application to image encryption with an improved aes algorithm. *Applied Sciences*, 11(3), 2021.

[69] D. Liu and Y. Wang. A dual-dimer method for training physics-constrained neural networks with minimax architecture. *Neural Networks*, 136:112–125, 2021.

[70] Y. Liu, X. Tong, and S. Hu. A family of new complex number chaotic maps

based image encryption algorithm. *Signal Processing: Image Communication*, 28(10):1548–1559, 2013.

[71] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[72] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations (ICLR)*, pages 1–17, 2019.

[73] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. Springer Science & Business Media, 2013.

[74] J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514, 2001.

[75] T. Matsubara, A. Ishikawa, and T. Yaguchi. Deep Energy-Based Modeling of Discrete-Time Physics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[76] D. McDuff and D. Salamon. *Introduction to Symplectic Topology*. Oxford University Press, 2017.

[77] X. Meng and G. E. Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics*, 401, 2019.

[78] C. Miehe, M. Hofacker, and F. Welschinger. A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits. *Computer Methods in Applied Mechanics and Engineering*, 199(45):2765–2778, 2010.

[79] S. Moon, J.-J. Baik, and J. M. Seo. Chaos synchronization in generalized lorenz systems and an application to image encryption. *Communications in Nonlinear Science and Numerical Simulation*, 96:105708, 2021.

[80] R. Mukherjee, Q. Li, Z. Chen, S. Chu, and H. Wang. Neuraldrop: Dnn-based simulation of small-scale liquid flows on solids, 2018.

[81] K. NAOE, H. TANAKA, and Y. TAKEFUJI. Information security techniques based on artificial neural network. *Journal of the Japanese Society for Artificial Intelligence*, 21(5):577–585, 2006.

[82] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, 7:19143–19165, 2019.

[83] J. M. Newby, A. M. Schaefer, P. T. Lee, M. G. Forest, and S. K. Lai. Convolutional neural networks automate detection for tracking of submicron-scale particles in 2d and 3d. *Proceedings of the National Academy of Sciences*, 115(36):9026–9031, 2018.

[84] A. N. Njah. Tracking control and synchronization of the new hyperchaotic liu system via backstepping techniques. *Nonlinear Dynamics*, 61(1-2):1–9, 2009.

[85] O. Noakoasteen, S. Wang, Z. Peng, and C. Christodoulou. Physics-informed deep neural networks for transient electromagnetic analysis. *IEEE Open Journal of Antennas and Propagation*, 1:404–412, 2020.

[86] M.-C. Pai. Global synchronization of uncertain chaotic systems via discrete-time sliding mode control. *Applied Mathematics and Computation*, 227:663–671, 2014.

[87] L. M. Pecora and T. L. Carroll. Synchronization of chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(9), 2015.

[88] M. Poli, S. Massaroli, A. Yamashita, H. Asama, and J. Park. Torchdyn: A neural differential equations library. *arXiv preprint arXiv:2009.09346*, 2020.

[89] A. Polyanin and V. Zaitsev. *Handbook of Nonlinear Partial Differential Equations*. Chapman and Hall/CRC, 2003.

[90] M. Preishuber, T. Hütter, S. Katzenbeisser, and A. Uhl. Depreciating motivation and empirical security analysis of chaos-based image and video encryption. *IEEE Transactions on Information Forensics and Security*, 13(9):2137–2150, 2018.

[91] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[92] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017.

[93] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations, 2017.

[94] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. In *Conference on Learning for Dynamics and Control (L4DC)*, 2020.

[95] K. Rath, C. G. Albert, B. Bischl, and U. von Toussaint. Symplectic Gaussian process regression of maps in Hamiltonian systems. *Chaos*, 31(5):053121, 2021.

[96] P. Ren, C. Rao, Y. Liu, Z. Ma, Q. Wang, J.-X. Wang, and H. Sun. Physics-informed deep super-resolution for spatiotemporal data, 2022.

[97] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.*, 108(5):058301, 2012.

[98] S. Sæmundsson, K. Hofmann, A. Terenin, and M. P. Deisenroth. Variational integrator networks for physically meaningful embeddings. In *Artificial Intelligence and Statistics (AISTATS)*, volume 108, pages 3078–3087, 2019.

[99] S. Saemundsson, A. Terenin, K. Hofmann, and M. Deisenroth. Variational integrator networks for physically structured embeddings. In *International Conference on Artificial Intelligence and Statistics*, pages 3078–3087. PMLR, 2020.

[100] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia. Hamiltonian Graph Networks with ODE Integrators. *arXiv*, 2019.

[101] H. Sano, M. Wakaiki, and T. Yaguchi. Secure Communication Systems Using Distributed Parameter Chaotic Synchronization. *Transactions of the Society of Instrument and Control Engineers*, 57(2):78–85, 2021.

[102] H. Sano, M. Wakaiki, and T. Yaguchi. Secure communication systems based on synchronization of chaotic vibration of wave equations. *Journal of Signal Processing*, 26(6):147–158, 2022.

[103] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. SchNet - a deep learning architecture for molecules and materials. *J. Chem. Phys.*, 148(24):241722, 2018.

[104] H. Scott Dumas. *KAM Story, The: A Friendly Introduction To The Content, History, And Significance Of Classical Kolmogorov-Arnold-Moser Theory*. World Scientific Publishing Company, 2014.

[105] K. Seetharaman and M. Jeyakarthic. Statistical distributional approach for scale and rotation invariant color image retrieval using multivariate parametric tests and orthogonality condition. *Journal of Visual Communication and Image*

Representation, 25(5):727–739, 2014.

[106] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[107] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[108] P. Shi, Z. Zeng, and T. Liang. Physics-informed convnet: Learning physical field from a shallow neural network, 2022.

[109] K. Shukla, P. C. D. Leoni, J. Blackshire, D. Sparkman, and G. E. Karniadakis. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks, 2020.

[110] I. Steinbach. Phase-field models in materials science. *Modelling and Simulation in Materials Science and Engineering*, 17(7):073001, 2009.

[111] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer Science & Business Media, 2008.

[112] S. Suri and R. Vijay. An aes–chaos-based hybrid approach to encrypt multiple images. In *Recent Developments in Intelligent Computing, Communication and Devices*, pages 37–43. Springer Singapore, 2017.

[113] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014.

[114] M. Tao. Explicit symplectic approximation of nonseparable hamiltonians: Algorithm and long time performance. *Phys Rev E*, 94(4-1):043303, 2016.

[115] M. Tariq, N. Memon, S. Ahmed, E. D. S. Tayyaba, T. Mushtaq, N. A. Mian, M. Imran, and M. Ashrf. A review of deep learning security and privacy defensive techniques. *Mobile Information Systems*, 2020:1–18, 2020.

[116] P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins. Hamiltonian Generative Networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[117] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, M. Walczak, J. Pfrommer, A. Pick, R. Ramamurthy, J. Garcke, C. Bauckhage, and J. Schuecker. Informed machine learning - a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions*

on Knowledge and Data Engineering, pages 1–1, 2021.

[118] L. Wang, Y. Wang, Z. Lin, J. Yang, W. An, and Y. Guo. Learning a single network for scale-arbitrary super-resolution, 2021.

[119] X. Wang and M. Zhao. An image encryption algorithm based on hyperchaotic system and dna coding. *Optics and Laser Technology*, 143:107316, 2021.

[120] M. Wei and X. Zhang. Super-resolution neural operator, 2023.

[121] A. A. Wheeler, B. T. Murray, and R. J. Schaefer. Computation of dendrites using a phase field model. *Physica D*, 66(1):243–262, 1993.

[122] J. D. Willard, X. Jia, S. Xu, M. S. Steinbach, and V. Kumar. Integrating physics-based modeling with machine learning: A survey. *ArXiv*, abs/2003.04919, 2020.

[123] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601(7894), 2022.

[124] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601(7894):549–555, 2022.

[125] Y. Wu. Npcr and uaci randomness tests for image encryption. *Cyber Journals: Journal of Selected Areas in Telecommunications*, 2011.

[126] S. Xiong, Y. Tong, X. He, C. Yang, S. Yang, and B. Zhu. Nonseparable symplectic neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[127] K. Xu and E. Darve. Adcme: Learning spatially-varying physical fields using deep neural networks, 2020.

[128] M. YAMATANI, Y. NAKAGAMI, H. FUKE, A. KAWACHI, M. KOZAI, Y. SHIMIZU, and T. YOSHIDA. New particle identification approach with convolutional neural networks in gaps. *Journal of Evolving Space Activities*, 1:9, 2023.

[129] J. Yang, S. Shen, H. Yue, and K. Li. Implicit transformer network for screen content image continuous super-resolution, 2021.

[130] H.-T. Yau, C.-L. Kuo, and J.-J. Yan. Fuzzy sliding mode control for a class of chaos synchronization with uncertainties. *International Journal of Nonlinear Sciences and Numerical Simulation*, 7, 2006.

[131] Y. Yu and H.-X. Li. Adaptive hybrid projective synchronization of uncertain chaotic systems based on backstepping design. *Nonlinear Analysis: Real World Applications*, 12(1):388–393, 2011.

[132] C. Yuhan, M. Takashi, and Y. Takaharu. Variational integrator for hamiltonian neural networks. *IEICE Proceeding Series*, 71:25–28, 2022.

[133] N. J. Zabusky and M. D. Kruskal. Interaction of "solitons" in a collisionless plasma and the recurrence of initial states. *Phys. Rev. Lett.*, 15(6):240–243, 1965.

[134] Y. D. Zhong, B. Dey, and A. Chakraborty. Dissipative SymODEN: Encoding Hamiltonian Dynamics with Dissipation and Control into Deep Learning. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations (DeepDiffEq)*, 2020.

[135] Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control. In *International Conference on Learning Representations (ICLR)*, 2020.

[136] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

Doctor Thesis, Kobe University
“Deep Learning for Modelling and Simulations in Physics” , 149 pages
Submitted on July 11th, 2023
When published on the Kobe University institutional repository /*Kernel*/, the publication date shall appear on the cover of the repository version.