Doctral Dissertation
博士論文

Exploration of Classical Integrable Systems Assisted by Neural Networks
（ニューラルネットワークによる古典可積分系の探索）

A Dissertation Submitted for the Degree of Doctor of Philosophy
December 2020
令和2年12月博士(理学)申請

Department of Physics, Graduate School of Science,
The University of Tokyo
東京大学大学院理学系研究科物理学専攻

Fumihiro Ishikawa
石川 文啓

Acknowledgments

This thesis is fully supported by Prof. Synge Todo. I would like to express my gratitude for his sincere supports. His insights made me notice some points I overlook due to being in a hurry. He patiently supervised me and discussed physics even if I often had a setback. His energetic attitude made me confront this research. Without his guidance and persistent help, this thesis would not have been possible.

I am also grateful to my collaborator, Assistant Professor Hidemaro Suwa. This research began with his advice when I lost my direction of research. Without his advice, I would not have come up with this theme and achieved the results. Furthermore, he encouraged me toward this thesis and gave me advice from his perspective. It provided me various approaches to this research.

I also appreciate the help of the other members of the Todo group. Project Lecturer Tsuyoshi Okubo gave me a lot of proper criticism of my presentations regularly. His advice improved my understanding of the topics. I would like to thank my colleagues, Takuya Yamamoto, Ken Nakanishi, Hayate Nakano, Chihiro Kondo, Ruixiao Cao, Yoshiki Maruyama, Tomoyuki Morishita, RuQing Xu, Taito Kutsuzawa, and Xun Zhao. I discussed various topics with them. They are not only practical subjects such as physics and computational science but also desultory topics such as hobbies. The conservations encouraged me and often produced some ideas to solve the trouble in the research.

I also extend my gratitude to all the people who have been involved with me, such as secretariats of the department of science, secretaries in the Todo group, alumni of the Todo group, and my friends. I would not have passed my research life without their supports.

Furthermore, I express my appreciation to Prof. Shingo Katsumoto as a mentor from the MERIT program. He encouraged me and kept my motivation toward the research. He also gave me the advice to attitude to study physics. His advice was indispensable and helped me to confront this research.

Finally, I would like to express the deepest appreciation to my parents Tokuji and Kiyomi Ishikawa, and my sister Momoko Ishikawa. Without their supports and encouragement, I would not have written this thesis. I express my heartfelt gratitude for their supports.

This calculation has been done using NVIDIA GPGPU at Institute for Physics of Intelligence ($i\pi$), the University of Tokyo. Furthermore, I acknowledge the support of the Materials Education program for the future leaders in Research, Industry, and Technology (MERIT).

Fumihiro Ishikawa

Abstract

Recently, machine learning techniques are significantly developed and applied to various fields. Especially, one of the machine learning techniques called "neural network" achieves prodigious success. In parallel with the success of machine learning, applications to fundamental research are studied energetically. In physics, various applications have also been reported. They include attempts to automatically extract the models and physical concepts from the data by the ability of neural networks. The approaches are expected to be prominently powerful but still developing. In the present thesis, we propose a method to construct the classical integrable systems by neural networks, which has been considered quite challenging so far. In previous studies, integrable systems have been found via ansatz of the Lax pair or assumptions for solutions of the equation of motion. Our method constructs integrable systems neither the Lax pair nor the assumptions of the solutions. Instead, we assume the Hamiltonian described by the action variables and find the natural Hamiltonian, which is consist of a kinetic term and a potential function. In other words, we construct the potential function exhibiting the classical integrability from the reduced Hamiltonian based on neural networks. The action variables are conserved quantities that characterize the integrable systems. We expect that method provides us unrevealed integrable systems that have not been discovered in previous studies. We represent the canonical transformation of the action variables and the potential function by neural networks. The canonical transformation is represented by a particular neural network constructed with invertible neural networks. Our neural network can express the canonical transformation for the Toda lattice, which has not been archived by the previous proposed neural networks. We also invent several novel techniques to achieve our purpose. We apply the adjoint method with the symplectic integrator to calculate the gradient efficiently. Moreover, we introduce some quantities evaluating the validity of the trained neural networks. We introduce various loss functions and structures of the neural networks and discuss appropriate combinations of them for the present purpose. In particular, we adopt two types of the loss function to find integrable systems. The first loss function consisting of just the loss of conservation of the action variables can find some integrable systems. The loss function of conservation of the action variables evaluates the time series generated by the potential function represented by the neural network. The second loss function consists of loss functions about conservation of the action variables, an equivalence of the energy, and the statistical properties. We demonstrate the loss function with the loss of the energy and the statistical properties find the desired integrable system through an application to the Toda lattice. As a result, we propose an appropriate construction of the loss function and neural networks for the exploration of the integrable systems. We also obtain the potential function which is the integrable system characterized by a reduced Hamiltonian with perturbations. By analyzing the obtained neural networks, we extract insights into candidates of the integrable systems and determine the analytical representation of the suggested integrable system.

Contents

Chapter 1

Introduction

In the past few decades, machine learning techniques are widely used and proven to be effective for many problems. One of the most famous machine learning techniques is called the “neural network.” Its applications are not restricted to regression problems but spread to vast complex theoretical and real issues. For instance, in the pattern recognition of images, the neural network can detect the edge of symbols automatically [1]. In the reinforce learning, the neural network suggests new strategies in the games, which are hard to come up with by human [2]. In natural language processing, the neural network learns the linguistic pattern and realizes the high accuracy translation [3]. For the application to generative models, the neural network learns the manifold of the data by extracting the features [4]. An essential point of the neural network method is to extract patterns from the vast data and provide new insights. The neural network has started from the perceptron model in 1958 [5]. It was a novel development in machine learning, though the representation power was low to apply to real problems at that moment. The ability of the perceptron model can be increased by multi-layering and widening the layers. It is proved that the multi-layered neural networks can approximate arbitrary functions in several cases [6–9]. On the other hand, the multi-layered neural networks require considerable numerical resources to optimize the parameters. In recent years, computational power increases drastically thanks to semiconductor technologies and parallel computing techniques. The computational power of a single processor has been growing according to Moore’s low and enables us to execute various tasks more quickly [10]. The parallel computing techniques, such as graphical processing units (GPU), provides us a way to treat the large matrix calculations [11, 12]. These advances in computer science raise new methods, such as backpropagation, calculation graph, and so on [13, 14]. Novel records of neural networks based on the learning with GPU are shown in the 2000s and subsequently located in an important position. Through such computational techniques, neural networks manifest power nowadays.

In parallel with the success of machine learning, applications to the fundamental research attract more attention. Needless to say, physics is not an exception. In recent years, a lot of applications in physics are reported [15]. For instance, machine learning methods have been applied to materials science. In this field, machine learning methods are used to explore new materials by extracting patterns from data. The techniques are called “materials informatics” [16]. In materials informatics, they use not only the neural network but also various statistical inference methods. The rich representation power and the automatic extraction of the features are the essential advantages of machine learning, especially neural networks. From this viewpoint, the application of neural networks to the trial functions is come up with. Particularly, the restricted Boltzmann machine (RBM) is used for the quantum physics [17–20]. In another direction, neural networks are applied to representing the potential function for the molecular dynamics and the first-principles calculations [21–24]. According to the success of neural networks in the classification task, there are applications to detect the phase transitions [25–29]. It is regarded as an application of the ability to connect tremendous data with a few desirable quantities. Along this direction, there are various applications. An example is the machine-learning parameter estimation, such as the model of the gravitational lens [30]. Another one is the modeling of the critical temperature of the superconductivity [31]. If we focus on the ability to extract features of the data, it is natural to come up with the approaches to discover the physical concepts and laws through making the models from the data. Some researchers have tried to make neural networks to find the analytical representations of the physical laws from the data [32]. Another approach is to learn features of the physics through the internal layers in the neural networks [33]. They are expected to emulate how to reduce the phenomena in the real systems to a simple law of physics as physicists do. For the other case, there are applications to various field of physics, such as Monte Carlo methods [34–41], equilibrium statistical physics [42–44], non-equilibrium statistical physics [45–47], designing experimental setups [48], photonic inverse design [49], analytic continuation [50], detection of anomaly data [51], and so on.

Within the above overview, it seems that there are mainly three directions for the applications of machine learning so far. The first direction is using machine learning for the acceleration of the simulations. For instance, as mentioned above, there are several reports about the acceleration of the Monte Carlo methods and the molecular dynamics calculation. The essential point is the combination of neural networks and the methods to correct the approximation introduced by the neural networks. The neural networks approximate or predict the genuine simulations by and reduce the computational costs. To produce unbiased results, the error due to the approximation is to be corrected. For the Mote Carlo methods, the balance condition and the reweighting method play such a role [35]. The second direction is to use neural networks as highly representable trial functions. This approach attracts more attention in the simulations of quantum many-body systems. As some theorems prove, the representation power of the neural networks has the potential to approximate the arbitrary functions [6, 7, 9, 20]. Besides other variational methods, the problems are how to optimize the trial functions and validate the results. The third direction is to extract new insights and models from the data by machine learning. As mentioned above, one of the advantages of machine learning, especially neural networks, is the ability to extract the features of the data. They also make the various models that connect the vast data to the quantities understandable for us. For instance, the neural network can detect the phase transition directly from the snapshots of configuration [28]. Furthermore, the machine learning predicts the properties of materials from the data and propose the desirable compounds to us. The problem with these approaches is that the limit of the application of the obtained models and insights is often unclear. The predictions or models can be validated by the prepared data but are not guaranteed for unknown data. One of the solutions is to understand the results and make the reasons for the obtained models.

The third direction is slightly different from the other two approaches. In the first two directions, machine learning is applied to replace some part of the existing numerical techniques. As the introduced approximation is corrected exactly afterward, it does not affect the whole validity of the calculation. For instance, in the Monte Carlo methods, the results become reliable by the reweighting and balance conditions. In the variational methods for quantum physics, the variational principle holds even if the trial functions are replaced. These applications are highly reliable because of the construction but limit the power of machine learning. In contrast, though the third direction has a little uncertainty, they can extract insights from the data. The insight are often difficult to be found directly by us. As mentioned above, neural networks can make the connection between the tremendous data and a few quantities. In fact, the neural network can detect the phase transition from the vast configurations directly. Through the task, the layers in the neural networks extract some physical quantities to detect the phase transition [29]. Furthermore, the neural network solves the celestial problems by constructing the relevant variables through [32, 33]. It finds the models from the data by the extreme processing capacity. In other cases, the materials informatics can extract some tendencies of properties of materials and propose desired one. The things we have to do are to give the desired quantities and the huge appropriate data to machine learning. The quantities play a role of the purpose of the learning. Once one prepares them, as far as the representation power is sufficient and the targets are appropriate, the machine learning is expected to achieve the purpose. As mentioned above, needless to say, we have to validate the results and models by interpreting the output. Although the procedure seems extremely different from the methodologies so far, physicists often perform similar activities. One of the representative examples is Johannes Kepler, who find the law of the planetary motion from the observation data by Tycho Brahe. In this sense, the aforementioned third direction of the applications is regarded as automation of finding models from the data. It means that the applications of machine learning provide us the mathematical abilities which are specialties for some researchers. As computers provide us the computation power that just some specialists have, the neural networks popularize the mathematical and sophisticated techniques. Furthermore, the methods are not only tools to automate finding processes, but augment our searching ability. It is expected that we find new systems and models which are difficult to be come up with by ourselves. From the point of view, we place importance on the progress of the third direction of the applications.

In the present thesis, we attempt to automate finding the Hamiltonian by neural networks in order to help the exploration of the integrable systems. Furthermore, we look for the appropriate protocol to use the neural networks for the third direction of applications.

The integrable systems are the systems that have many independent conserved quantities. There are several definitions of the "integrability" in classical physics and the quantum physics [52]. The integrable systems appear in the various field of physics, such as the BCS Hamiltonian and the Bose gas [53–55]. In the mathematical aspects, there are many fields related to the integrable systems [52, 56]. The essential point is that many conserved quantities restrict the motions. The restriction causes rich phenomena and raises various questions. Recently, one of the fundamental problems, thermalization of the integrable system, attracts more attention [57]. From the viewpoint of thermodynamics, a system is expected to relax to the equilibrium state. For the integrable system, however, the system does not relax to the state described by the ordinary equilibrium ensemble, but to different stable states due to the restriction of motions by the many conserved quantities. The state is not described by the ordinary Gibbs ensemble, but the generalized Gibbs ensemble (GGE) [58, 59]. The phenomena raise questions about the fundamental aspects of thermodynamics. Moreover, integrability also affects the relaxation dynamics of near-integrable systems. Due to the perturbations, the near-integrable systems finally relax to the state described by the Gibbs ensemble. However, the systems exhibit the meta-stable state characterized by the GGE, which is called "prethermal state" [57, 60]. The phenomenon is not just theoretical but realized by the experiment [55]. Although the discussion of the prethermalization of the near-integrable systems is progressed in quantum physics energetically, the same problems are known in classical physics. In the simulations, it is known that the relaxation by the nonlinear lattice model exhibits the recurrence motions contrary to the expectations. The model is constructed by a harmonic oscillator chain with nonlinear interactions. The phenomenon seems to indicate the nonlinear perturbation does not cause the relaxation to the equilibrium state. The problem is called the Fermi-Pasta-Ulam-Tsingou paradox [61, 62]. Nowadays, the phenomenon is regarded as the prethermalization in the perturbed Toda lattice [62–66]. In parallel with the GGE in quantum physics, the GGE of the Toda lattice is also proposed recently [67–69].

The construction of integrable systems has helped the understanding of their fundamental aspects. Currently, there are different approaches to construct integrable systems. One of the conventional approaches is the heuristic approach. The most famous integrable system, the Toda lattice, is found by assuming the solutions of the equation of motion [70, 71]. The potential function consists of the exponential function. Toda comes up with the periodic solutions of the equation excepting the trigonometric functions and makes the potential for consistency. The solutions describe the solitary wave, called "soliton," which does not collapse with the time development. Another approach is assuming the ansatz of the Lax pair. The Lax pair is a pair of matrices equivalent to the original equation of motion. If one can construct the pair, enough conserved quantities are produced systematically. In other words, if one constructs the Lax pair transformable to the Hamiltonian equation, an integrable system is found. By this approach, a group of integrable systems is found, which is "Calogero-Moser systems" [72–74]. The construction is generalized in terms of the Lie group [52, 56, 75]. From the viewpoint of differential geometry, several integrable systems are constructed by the projection of the higher-dimensional manifolds [76]. The higher-dimensional manifolds are represented by the Lie group or the differential manifold. The equations of motion of such integrable systems are realized by the geodesic equation on the higher-dimensional manifolds. However, it is currently unclear that all of the integrable systems can be understood by the geodesic motion in the higher-dimensional space.

Furthermore, there are some approaches, which are possible in theory but hard practically. We focus on the integrable lattice systems described by the Hamiltonian mechanics, such as the Toda lattice and Calogero-Moser systems. One of the approaches is to assume the Hamiltonian described by the action variables. Here, we call such Hamiltonian "transformed Hamiltonian." Theoretically, an integrable system that exhibits the bounded motion has a particular canonical coordinate called "action-angle variables" [77]. The action-angle variables consist of the action variables and the conjugate coordinates called the angle variables. If we consider the periodic motion in the phase space, the action variables correspond to the radii of the periodic motion, and the angle variables correspond to the angle. It is also known that the transformed Hamiltonian is described by just the action variables. Therefore, there is a canonical transformation to the variables. As an extreme statement, one already knows the integrable system when one can make the transformed Hamiltonian. However, a realistic Hamiltonian should consist of the kinetic term and the potential function. A Hamiltonian of this form is called "natural Hamiltonian" [76]. Thus, the construction of the integrable system from the transformed Hamilton is equivalent to construct the canonical transformation of the action-angle variables and the potential function in a consistent way. In general, however, it is hard to find the transformation except for trivial cases. The canonical transformation includes complex operations on the canonical coordinates. For instance, the canonical transformation of the Toda lattice is obtained from the Lax pair generated by the original Hamiltonian. Therefore, there is a probability that the unrevealed integrable systems are found by the approach if it is realized. As the Toda lattice casts light on the soliton and the rich mathematical aspects, the unrevealed integrable system might have the key to new fields.

In the present thesis, we propose an approach by using neural networks to find integrable systems. This approach is introduced by us and enables us to search the unrevealed systems which are difficult for the previous methods for the first time. Comparing with previous researches, our concept is different from them. In previous researches, their aim is just to lean the dynamics and properties from the experimental data or the data generated from known systems. In contrast, the concept of our approach is to find systems which have desired properties. In this sense, our approach has novelty. We represent a canonical transformation and a potential function by neural networks. Recently, some methods to represent the canonical transformation by neural networks are reported [78, 79]. We call the canonical transformation represented by the neural network "neural canonical transformation," following the previous research [78]. Furthermore, we call the potential function represented by the neural network "neural potential function" in this thesis. In particular, our canonical transformation is represented by "normalizing flow", a neural network representing invertible transformations. The normalizing flow is a famous method attracting much attention recently for the neural generative model [80]. By these representations by neural networks, we find a natural Hamiltonian corresponding to a given transformed Hamiltonian. In this sense, this approach corresponds to the aforementioned third direction of applications of neural networks. In other words, we attempt to automate finding a Hamiltonian of an integrable system. Concretely, we create the training data by sampling from the Boltzmann distribution characterized by the transformed Hamiltonian. Thus, we can define the temperature of the data. The temperature enables us to estimate the reliable region of the neural potential function. We originally construct a loss function by combining the energy, conservation of the action variables, and the statistical properties. In particular, the loss function for the action variable conservation evaluates the time series generated by the neural potential function. For the efficient calculation of gradients of the loss function, we use the adjoint method with a symplectic integrator [81, 82]. We introduce two scores to validate the neural potential function and the neural canonical transformation. The score evaluates the conservation of the action variables obtained by the neural canonical transformation from the time series generated by the neural potential function. If the action variables are conserved during the time development, the obtained potential function is expected to be close to the integrable system. It is noted that the obtained potential functions do not produce the perfect integrable systems but some approximations. In other words, we obtain integrable systems with small perturbations. Thus, the learning results provide candidates of the integrable systems corresponding to the given transformed Hamiltonian. In this sense, the neural networks assist us in finding new integrable systems by approximation.

The rest of this thesis is organized as follows: In Chap. 2, we give a brief review of the classical integrable systems. In Chap. 3, we review the basic concepts of neural networks and some recent progress used in our approach. In Chap. 4, we introduce some realization of our approach and compare the methods by applying them to the Toda lattice. Moreover, we attempt to find the integrable system of the transformed Hamiltonian by the proposed method and discuss the results. In Chap. 5, we summarize the present thesis with an outlook.

Chapter 2

Classical Integrable Systems

2.1 Hamiltonian Mechanics

2.1.1 Formalism

We make a brief review of the classical integrable systems [77]. Furthermore, we introduce some basic points of canonical transformations to understand the structure of the neural networks proposed in the following chapters. To begin with, we review the Hamiltonian mechanics.

We note that we denote a vector as unbold symbol *e.g.* $x = (x_1, \dots, x_N)$ unless the representation causes confusion. Let a system be constructed by N particles and t_0, t_f be the initial time and final time, respectively. When canonical coordinates (q, p) and the Hamiltonian H are given, the action of the dynamics is defined as

$$S = \int_{t_0}^{t_f} \left(\sum_{i=1}^{N} p_i \dot{q}_i - H \right) dt, \tag{2.1}$$

and the equation of the motion is obtained by the variation:

$$\begin{aligned} &\delta \int_{t_0}^{t_f} \left(\sum_{i=1}^{N} p_i \dot{q}_i - H \right) dt = 0, \\ &\int_{t_0}^{t_f} \sum_{i=1}^{N} \left[\left(\frac{dq_i}{dt} - \frac{\partial H}{\partial p_i} \right) \delta p_i + \left(\frac{dp_i}{dt} + \frac{\partial H}{\partial q_i} \right) \delta q_i \right] dt = 0, \end{aligned} \tag{2.2}$$

where δq_i and δp_i are the variation of the positions and momentums respectively. It is called "Hamilton's principle." As a result, the dynamics of the system obeys the following equations:

$$\begin{aligned} \frac{dq_i}{dt} &= \frac{\partial H}{\partial p_i}, \\ \frac{dp_i}{dt} &= -\frac{\partial H}{\partial q_i}. \end{aligned} \qquad (i = 1, \cdots, N) \tag{2.3}$$

The r.h.s. of Eq. (2.3) is regarded as an operator acting on the coordinates. From the point of view, the classical Hamiltonian equations are represented as

$$\frac{dx}{dt} = D_H x = \Omega \nabla_x H, \tag{2.4}$$

where x is the canonical coordinates (q, p), D_H is the vector field associated with the Hamiltonian, so-called "Hamiltonian vector filed":

$$D_H := \sum_{i=1}^{N} \left[\frac{\partial H}{\partial p_i} \frac{\partial}{\partial q_i} - \frac{\partial H}{\partial q_i} \frac{\partial}{\partial p_i} \right], \tag{2.5}$$

and Ω is a $2N \times 2N$ block matrix defined as

$$\Omega := \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}, \tag{2.6}$$

where I is a $N \times N$ identity matrix. This vector field is also called the "Liouville operator" and represented as $\mathcal{L}$ [83]. By using the above terminology, the time development of the canonical coordinates is described as follows:

$$\begin{aligned} x(t) &= e^{(t_f - t_0)D_H} x(t_0), \\ e^{tD_H} &:= I + tD_H + \frac{1}{2}t^2 (D_H)^2 + \cdots , \end{aligned} \tag{2.7}$$

where e^{tD_H} is an exponential map induced by the vector field. This representation is useful for extending symplectic integrators, which is one of the numerical integration methods. The detail is reviewed in Appendix. A.1.

2.1.2 Canonical transformation

A canonical transformation is a map from canonical coordinates to other canonical coordinates. Let a transformation be a canonical transformation $f : x = (q, p) \longrightarrow z = (Q, P)$. If vector x obeys the Hamiltonian mechanics, transformed canonical coordinates z satisfy

$$\frac{dz}{dt} = J_f^T \Omega J_f \nabla_z H(x(z)), \tag{2.8}$$

where J_f is the Jacobian matrix of f and J_f^T is its transposed matrix. The equation of motion of z has the Hamiltonian form if and only if

$$J_f^T \Omega J_f = \Omega. \tag{2.9}$$

The matrix which satisfies Eq. (2.9) is called a symplectic matrix. It is easily understood that a composition of canonical transformations is also a canonical transformation. This is because the Jacobian matrix of the composition is a product of the Jacobian matrix of each function. Needless to say, the exponential map defined by Eq. (2.7) satisfies the condition and is also a canonical transformation. It means that the canonical transformation conserves the volume of the phase space:

$$\int \prod_{i=1}^{N} dq_i dp_i = \int \prod_{i=1}^{N} dQ_i dP_i \text{ or } \sum_{i=1}^{N} dq_i \wedge dp_i = \sum_{i=1}^{N} dQ_i \wedge dP_i, \tag{2.10}$$

where the second equation is in the differential form representation, which is equivalent to the first one. The representation is called the symplectic form. In terms of the differential form, the canonical transformation preserves the symplectic structure. It implies that the time development generated by the Hamiltonian vector field conserves the volume of the phase space. This is called the "Liouville theorem" [83]. It indicates the generalization of canonical transformations induced by the Lie exponential map [77]. Although there are various canonical transformations, we introduce two representative types in the following.

The first type is the canonical transformation defined by generating functions. The Hamiltonian equation of z is equivalent to the equations of x according to Eq. (2.9). Thus, the original equation of x and the equation of z satisfy Hamilton's principle simultaneously:

$$\delta \int_{t_0}^{t_f} \left(\sum_{i=1}^{N} p_i \dot{q}_i - H - \sum_{i=1}^{N} P_i \dot{Q}_i + K \right) dt = 0, \tag{2.11}$$

where $K = K(z)$ is the Hamiltonian of z and equivalent to $H(x(z))$. Since the variational of the action depends only on the endpoints, the integrand is given by the derivative of a smooth function $W(t)$. Hence, the canonical coordinates, x and z, are related via the following equation:

$$\sum_{i=1}^{N}(p_i dq_i - P_i dQ_i) - (H - K)dt = dW. \tag{2.12}$$

The l.h.s. of the equation is nothing but the total derivative of a certain function. Thus, one obtain p_i, P_i, K from a generating function as

$$\begin{aligned} &W_1 = W_1(q, Q, t) \\ &p_i = \frac{\partial W_1}{\partial q_i},\ P_i = -\frac{\partial W_1}{\partial q_i},\ K = H + \frac{\partial W_1}{\partial t}. \end{aligned} \tag{2.13}$$

In the transformation, q_i and Q_i are the independent variables and p_i, P_i are derived from the generating function. It is not hard to consider another pair of the independent variables, that are obtained by the Legendre transformation:

$$\begin{aligned} &W_2 = W_2(q, P, t) \text{ s.t. } dW_2 := d(W_1 + \sum_{i=1}^{N} P_i Q_i) \\ &p_i = \frac{\partial W_2}{\partial q_i},\ Q_i = -\frac{\partial W_2}{\partial P_i},\ K = H + \frac{\partial W_2}{\partial t}. \end{aligned} \tag{2.14}$$

The formulation indicates that there are four patterns for the pair of the independent variables. It is easy to prove that all the canonical transformations satisfy Eq. (2.9). For a concrete example, we construct a generating function induced by point transformations acting on the position. Let f be a point transformation from q to Q denoted as $Q_i = f_i(q)$. Through the chain rule of the derivative, the Jacobian matrix of the transformation from $\dot{Q}$ to $\dot{q}$ is obtained as

$$\frac{\partial \dot{Q}_i}{\partial \dot{q}_j} = \frac{\partial f_i}{\partial q_j}. \tag{2.15}$$

Because of the representation of the momentum in the Lagrangian formalism, the relation between p and P is easily derived as

$$p_i = \sum_{j=1}^{N} \frac{\partial f_j}{\partial q_i} P_j. \tag{2.16}$$

Thus, the generating function of the canonical transformation W is

$$\begin{aligned} &W = W(q, P),\ \text{s.t. } W = \sum_{i=1}^{N} P_i f_i(q), \\ &p_i = \sum_{i=j}^{N} P_j \frac{\partial f_j}{\partial q_i},\ Q_i = f_i(q),\ K = H. \end{aligned} \tag{2.17}$$

Second, we introduce the symplectic linear transformation. The transformation is defined as a symplectic matrix S:

$$z = Sx,\ \text{s.t. } S^T \Omega S = \Omega. \tag{2.18}$$

The simplest representation of the matrix is the Lie group generated by the Lie algebra:

$$S = e^{tX_{\mathfrak{sl}}} \tag{2.19}$$

$$X_{\mathfrak{sl}} = \begin{pmatrix} A & B \\ C & -A^T \end{pmatrix}, \tag{2.20}$$

where B and C are symmetry matrices and t is a scalar parameter. There are several factorizations of the symplectic matrices [84]. Generally, symplectic linear matrices are decomposed into three matrices. The factorization is called the Iwasawa decomposition [85]. Through the representation of the decomposition, the symplectic matrices are parameterized by the following elements:

$$\begin{aligned} S &= \mathcal{N}\mathcal{A}\mathcal{K}, \\ \mathcal{K} &= \begin{pmatrix} X & Y \\ -Y & X \end{pmatrix}, \ X + iY \in U(n) \\ \mathcal{A} &= \begin{pmatrix} D & 0 \\ 0 & D^{-1} \end{pmatrix}, \\ \mathcal{N} &= \begin{pmatrix} L & 0 \\ N & (L^{-1})^T \end{pmatrix}, \ L^T N = N^T L, \end{aligned} \tag{2.21}$$

where D is a positive diagonal matrix and L is a lower triangular with uniform diagonal elements. One can write the three factors in the opposite sequence by conjugating the matrices. If one replaces D with the semi-positive definite symmetry matrix and L with an identity matrix, it is nothing but the pre-Iwasawa decomposition [84, 86]. It means that the Iwasawa decomposition is the modification of the pre-Iwasawa decomposition so that each factor is taken from a subgroup of the symplectic group.

2.2 Integrable Systems

2.2.1 Definition of integrability

There are several definitions of integrability [52]. In the present thesis, we focus on the conventional definition, which is called the “Liouville-Arnold integrability” [75, 77]. To begin with, we introduce “involution” to mention the definition of the Liouville-Arnold integrability. Let N be the dimension of the position in a system. It is equivalent that the dimension of the phase space is $2N$. The definition of involution is that: N smooth functions of a Hamiltonian system denoted as ${F_i}_{i=1}^N$ are in involution if all the pairs of the functions satisfy the condition

$$\{F_i, F_j\} = 0, \qquad i, j = 1, \ldots, N, \tag{2.22}$$

where $\{\cdot\}$ is the skew-symmetric bilinear operator, called “Poisson brackets”, defined as

$$\{A, B\} := \sum_{i=1}^{N} \left[\frac{\partial A}{\partial p_i}\frac{\partial B}{\partial q_i} - \frac{\partial A}{\partial q_i}\frac{\partial B}{\partial p_i} \right]. \tag{2.23}$$

Here, let the N smooth functions are the first integrals, which means the functions are conserved quantities. One of the functions can be the Hamiltonian itself. Then, the definition of the “Liouville integrability” is that: if the Hamiltonian system has N first integrals in involution and they are independent at every point of the phase space, the system has the Liouville integrability and is called completely integrable in the Liouville sense. Here, the independence of the functions means that dF_i are linearly independent, *i.e.* the rank of the Jacobian matrix of F_i is equal to N.

If the system has integrability, several properties exist. One of the most important properties is that the trajectories are diffeomorphic to the N-dimensional torus if the motions are bounded. Let $\theta_i, i = 1 \ldots N$ be the angular coordinates of each dimension of the torus respectively, and F_i be the first integral of the system. Then, the motion of the integrable system is represented by

$$\frac{d\theta_i}{dt} = \omega_i(F), \tag{2.24}$$

where $\omega_i(F)$ are determined by the value of F_i. In general, the first integrals are not canonical coordinates. It is known that, however, there are functions of F, denoted as $\{I_i\}_{i=1}^N$, which are the canonical coordinates with θ_i if the motions are periodic. The coordinates are called "action-angle variables":

$$\sum_{i=1}^{N} dq_i \wedge dp_i = \sum_{i=1}^{N} dI_i \wedge d\theta_i, \tag{2.25}$$

and the equation of motion is

$$\begin{aligned} \frac{dI_i}{dt} &= 0, \\ \frac{d\theta_i}{dt} &= \frac{\partial H(I)}{\partial I_i} = \omega_i(I). \end{aligned} \tag{2.26}$$

In this sense, the condition where the system has the action-angle variables is equivalent to the definition of the Liouville-Arnold integrability [77]. We note that the action-angle variables correspond to the periodic motion, *e.g.* rotation and oscillation. For instance, the total momentum of the free particles is not transformed into the action-angle variables. The action-angle variables are readily obtained if the Hamiltonian is separable to the N decoupled Hamiltonians:

$$H = \sum_{i=1}^{N} H_i(q_i, p_i), \tag{2.27}$$

where H_i is the Hamiltonian for each particle. We consider that each motion generated by each partial Hamiltonian H_i is periodic. Even if the motion is the rotation or the oscillation, the trajectory in the phase space is closed and the following quantity is conserved:

$$I_i = \frac{1}{2\pi} \oint p_i dq_i, \tag{2.28}$$

where the contour integration is carried out on the trajectory. This quantity is also proved to be a canonical coordinate. The generating function W is given by

$$\begin{aligned} &W(q, I) = \sum_{i=1}^{N} W_i(q_i, I_i), \; W_i(q_i, I_i) = \int^{q_i} p_i(q_i', I_i) dq_i', \\ &p_i = \frac{\partial W}{\partial q_i}, \; \theta_i = -\frac{\partial W}{\partial I_i}. \end{aligned} \tag{2.29}$$

The generating function corresponds to the second generating function defined by Eq. (2.14). The transformation satisfies the condition Eq. (2.9).

One of the simplest examples is a harmonic oscillator chain constructed by N particles with periodic boundary conditions. It is well-known that the system can be transformed to the $N-1$ independent harmonic oscillators and the free motion of the center of mass:

$$H = \frac{1}{2N}\left(\sum_{i=1}^{N} p_i\right)^2 + \sum_{k=1}^{N-1}\left[\frac{P_k^2}{2} + \frac{\omega_k^2 Q_k^2}{2}\right], \; \omega_k := \sqrt{4J}\sin\left(\frac{\pi k}{N}\right), \tag{2.30}$$

where ω_i is the frequency characterized by the dimensionality of the chain, and J is the coupling constant, and P, Q are coordinates decoupling the interactions. Here, one can introduce the conserved quantities, which mean the "radii" of the trajectory in the phase space:

$$\begin{aligned} I_k &= \frac{P_k^2}{2\omega_k} + \frac{\omega_k Q_k^2}{2}, \; k = 1, \dots N-1, \\ I_0 &= \frac{1}{2N}\left(\sum_{i=1}^{N} p_i\right)^2. \end{aligned} \tag{2.31}$$

It is obvious that I_0 always exists when the potential function depends only on the difference of the positions. Since the transformation from the original coordinates to the decoupled oscillators is a canonical transformation, it is easy to prove that the quantities are N linearly independent first integrals in involution. The angular variables are defined by

$$\theta_k = \tan^{-1}\left(\frac{P_k}{\omega_k Q_k}\right) \; k = 1, \dots N-1. \tag{2.32}$$

For convenience, we focus on just one pair of the coordinates. We denote the index as k. Then, the Jacobian matrix is given by

$$\begin{pmatrix} \frac{\partial I_k}{\partial P_k} & \frac{\partial I_k}{\partial Q_k} \\ \frac{\partial \theta_k}{\partial P_k} & \frac{\partial \theta_k}{\partial Q_k} \end{pmatrix} = \begin{pmatrix} \frac{P_k}{\omega_k} & \omega_k Q_k \\ \frac{\omega_k Q_k}{P_k^2+\omega_k^2 Q_k^2} & \frac{-P_k}{P_k^2+\omega_k^2 Q_k^2} \end{pmatrix}. \tag{2.33}$$

It is straightforward to confirm that this Jacobian matrix satisfies the condition, Eq. (2.9). Consequently, the harmonic oscillator chain is completely integrable and has the action-angle variables, and the Hamiltonian is given by

$$H = I_0 + \sum_{k=1} \omega_k I_k. \tag{2.34}$$

We can also obtain the action variable from the definition, Eq. (2.28). We denote each energy of the periodic motion as

$$E_k = \frac{P_k^2}{2} + \frac{\omega_k^2 Q_k^2}{2}. \tag{2.35}$$

Then, the contour integration in Eq. (2.28) is given by

$$\begin{aligned} I_k &= \frac{1}{2\pi} \oint P_k dQ_k \\ &= \frac{1}{\pi} \int_{-\frac{\sqrt{2E_k}}{\omega_k}}^{\frac{\sqrt{2E_k}}{\omega_k}} \sqrt{2E_k - \omega_k^2 Q_k^2 dQ_k} \\ &= \frac{E_k}{\omega_k}, \end{aligned} \tag{2.36}$$

which is nothing but the quantities defined by Eq. (2.31). As well as the ordinary canonical coordinates, the action-angle variables are not unique due to the translation of the origin. Furthermore, the angle variables can be transformed by a unimodular matrix [87].

According to the above observations, there are some trivial integrable systems. The first one is the one-dimensional one-particle system, where the dimension of the phase system is just two. The system has a trivial conserved quantity, the energy. Thus, the system whose dimension of the phase space is two is always integrable. The second system is the system constructed by the independent subsystems whose phase space dimension is two. In other words, the subsystems are decoupled from each other. As mentioned above, each subsystem is integrable. Thus, the system consisting of the integrable system is also integrable.

2.2.2 Construction of first integrals: Lax pair

Except for the trivial cases, in general, it is hard to find the first integrals from the equation of motion. One of the representative methods to construct the first integrals is to find the Lax pair [88]. It is also frequently called the isospectral deformation method. The Lax pair is a pair of two matrices, denoted as L and M, obeying the following equation of motion:

$$\frac{dL}{dt} = [M, L], \tag{2.37}$$

where $[\cdot]$ denotes the anticommutator. The entries in the matrices depend on the canonical coordinates, and Eq. (2.37) is equivalent to the original Hamiltonian dynamics. By introducing the matrix u, which is driven by M, the evolution of L is represented as

$$L(t) = u(t)L(0)u^{-1}(t),\ u(t)\ \text{s.t.}\ \frac{du}{dt} = Mu. \tag{2.38}$$

Since this equation is regarded as a similarity transformation, the eigenvalues of L is time-independent. This fact indicates that the Lax form can produce various conserved quantities, *e.g.* the trace of any power of L:

$$\text{trace}\left(L^k\right),\ k > 0. \tag{2.39}$$

By several methods, it is proved that the first integrals are in involution [56, 89, 90]. If the action-angle variables are found, the Lax pair is constructed from them by the following procedure. Let $\{D_i, E_i\}_{i=1}^N$ be a Lie algebra and they have the relations:

$$\begin{aligned} [D_i, D_j] &= [E_i, E_j] = 0, \\ [D_i, E_j] &= 2\delta_{ij}E_j. \end{aligned} \tag{2.40}$$

Then, the Lax pair is obtained as

$$\begin{aligned} L &= \sum_{i=1}^N \left(I_i D_i + 2I_i\theta_i E_i\right), \\ M &= \sum_{i=1}^N \left(\omega_i(I)E_i\right). \end{aligned} \tag{2.41}$$

where $\{I_i, \theta_i\}_{i=1}^N$ are the action-angle variables [91].

The Lax pair is not uniquely determined, and L is not necessary to be symmetric. A famous example is a Lax pair for the Toda lattice. Historically, the first Lax pair of Toda lattice is proposed through the analysis of the KdV equation [92]. After that, another representation is provided through the ansatz approach to find integrable systems [93]. Furthermore, the representations are different from the Lax pair constructed by the action-angle variables according to Eq. (2.41).

2.2.3 Many-body systems

There are several integrable systems [76]. Here, we introduce the two representative integrable models, the Toda lattice and the Calogero-Moser model [70–74].

The Toda lattice is defined by

$$H = \frac{1}{2m}\sum_{i=1}^N p_i^2 + \sum_{i=1}^N J\left(e^{-\alpha(q_{i+1}-q_i)} + \alpha(q_{i+1} - q_i) - 1\right), \tag{2.42}$$

where J and α are the coupling constants, and the system obeys the periodic boundary conditions. Since it is instructive to find the Toda potential function heuristically, we review the procedure given in Appendix B.2.1. The Toda lattice is also one of the soliton systems, and there are several approaches to solve the equation. There is another construction of the solution in terms of the soliton called the "Hirota direct method" [94]. By this method, one can construct the solutions corresponding to the multi-soliton states. The solutions are, however, realized in the infinite chain. There is a method to find the rigorous solution of a finite lattice by the Riemann geometry [95]. According to the definition of completely integrable systems, the Toda lattice also has the action-angle variables. The variables are obtained by the Lax pair, and the approach is called the "inverse scattering method" [96, 97]:

$$I_k = \frac{2}{\pi}\int_{\lambda_{2k}}^{\lambda_{2k+1}} \cosh^{-1}\left[\frac{(-1)^k}{2}\Delta(\mu)\right] d\mu, \tag{2.43}$$

where $\{\lambda_{2k}\}_{k=1}^{N-1}$ are the eigenvalues of the two matrices related to L in the Lax pair. A part of them is obtained from the standard Lax matrix defined by Eq. (B.31). The other is obtained from the matrix denoted by L^-, which is the Lax matrix defined by Eq. (B.31) except for L^-_{1N} and L^-_{N1}. The two elements are defined as $L^-_{N1} = -L_{N1}$ and $L^-_{1N} = -L_{1N}$, respectively. Δ in Eq. (2.43) is the function related to the determinant called the "discriminant" defined by

$$\det(L - \lambda I) = (-1)^N \left(\prod_{j=1}^{N} a_j \right) (\Delta(\lambda) - 2)\,, \tag{2.44}$$

where a_j is an element of the Lax matrix and I is an identity matrix. For numerical calculation, the evaluation of the discriminant is too heavy to iterate by the naive methods. However, there are efficient calculation methods, which are reviewed in Appendix. B.2.3. The action variables play a role of the amplitude of the soliton [98]. Concretely, there are N action variables and N angle variables, and they characterize the soliton motions. For instance, the action variable corresponding the "wave number" k describes the $k \mod N/2$ soliton motion. In this sense, the action variables of the Toda lattice are a nonlinear generalization of the normal modes in the harmonic oscillator. Furthermore, the action variables are approximately equal to the amplitude of the normal mode of a simple harmonic oscillator chain when the energy of the system is sufficiently low. For the Toda potential, the analytical representation of Hamiltonian $K(I)$ is not explicitly written down. The lowest order expansion is derived as [99, 100]

$$K(I) = I_0 + \sum_{i=1}^{N-1} \omega_i I_i + \frac{1}{4N} \sum_{i=1}^{N-1} I_i^2 + O(I^3), \tag{2.45}$$

where ω_i is the frequency defined by $2\sin\left(\frac{\pi i}{N}\right)$.

The second integrable model is the Calogero-Moser system. Calogero discovers a group of the classical integrable systems by assuming an ansatz of Lax pair [72–74]. The ansatz is represented as

$$\begin{aligned} L_{jk} &= \delta_{jk} p_j + (1 - \delta_{jk}) \alpha(q_j - q_k), \\ M_{jk} &= \delta_{jk} \sum_{l=1, l\neq j}^{N} \beta(q_j - q_l) - (1 - \delta_{jk}) \alpha'(q_j - q_k), \end{aligned} \tag{2.46}$$

where $\alpha'(x)$ is the derivative of $\alpha(x)$. The functions are assumed to be odd functions here, but this assumption can be abandoned to obtain more general solutions. These functions are determined by substituting them into the equation of motion Eq. (2.37). The relations are obtained as

$$\alpha'(y)\alpha(z) - \alpha(y)\alpha'(z) = \alpha(y + z)(\beta(y) - \beta(z)), \tag{2.47}$$

and the Hamilton equation is given by

$$\begin{aligned} \frac{dq_i}{dt} &= p_i, \\ \frac{dp_i}{dt} &= \sum_{k\neq i}^{N} \frac{\partial}{\partial q} \alpha^2(q_i - q_j). \end{aligned} \tag{2.48}$$

Comparing the Hamilton equation, the potential function is obtained as $V(q) = -\alpha^2(q) + \text{const}$. Solving Eq. (2.47), we obtain the reduced relation:

$$\beta(q) = \frac{\alpha''(q)}{2\alpha(q)}, \tag{2.49}$$

and the solutions of $\alpha(q)$. Consequently, the potential functions satisfying the condition are obtained as

$$V(q) = \begin{cases} q^{-2}, \\ a^2 \sinh^{-2}(aq), \\ a^2 \sin^{-2}(aq), \\ a^2 \mathscr{P}(aq), \end{cases} \tag{2.50}$$

where $\mathscr{P}(q)$ is the Weierstrass function having two periods. It is obvious that the lattice structure is fully connected. With the analysis by the Lie group, it is revealed that the form of the potential function depends on the lattice structure [93, 101]. The first type of potential function is called the "rational Calogero-Moser" system. Meanwhile, the most famous potential function known as the "Calogero-Moser" system is derived from another ansatz, which is one of the generalization. The generalization is to add the external field to the potential function [102, 103]. As a results, the new solutions are obtained as

$$\begin{cases} V(q) = q^{-2}, \ W(q) = bq^2 + cq^4, \\ V(q) = a^2 \sinh^{-2}(aq), \ W(q) = b\cosh(4aq + c), \end{cases} \tag{2.51}$$

where $W(q)$ is the external filed acting on each site. The Calogero-Moser system corresponds to the first equation and the Hamiltonian is given by

$$H = \sum_{i=1}^{N} \left(\frac{p_i^2}{2} + \omega^2 q_i^2 \right) + \sum_{i \neq j} \frac{J}{(q_i - q_j)^2}, \tag{2.52}$$

where J is the coupling constant.

2.2.4 Geometrical aspects

There is some geometrical approach to understand the integrable systems [75, 76]. In particular, the relatively new framework is the construction by a projection of the higher-dimensional manifolds. Several integrable systems are realized by the geodesic trajectory of a free particle on the higher-dimensional manifolds [104]. The manifolds are not only the Riemann manifolds but also the manifolds described by the Lie groups [93]. In the approach, some integrable systems are regarded as the projection of the geodesics on the Lie group. Recently, the projection of the geodesic trajectory on the higher dimensional manifold is related to the geodesics on the Lie group. This approach enables the Toda lattice to be generalized [101]. Furthermore, the Lie group approach is generalized to elucidate the existence of the wider class of the integrable systems, which is called the Adler-Kostant-Symes (AKS) theorem [56, 105–107].

Chapter 3

Neural Networks

3.1 General Description

There are many types of machine learning methods using the "neural network" concepts, such as recurrent neural networks, convolutional neural networks, generative adversarial networks, and so on [8, 13, 108, 109]. Furthermore, the new and various models are being developed and proposed energetically. Thus, it is hard to list all types of neural networks and introduce them. In this section, we focus on one of the representative neural networks, which is called the "feedforward neural networks" with dense layers.

Here, we consider the two vectors, $y = \{y_i\}_{i=1}^N$ and $x = \{x_i\}_{i=1}^N$. The purpose of the neural network is to obtain the approximation of some target function $y = f^*(x)$. One of the simplest approximations is to represent the function by linear combinations of some basis functions:

$$y_i = \sum_{j=1} w_{ij}\phi_j(x_j), \tag{3.1}$$

where $\{\phi_j\}$ are the basis functions and $\{w_{ij}\}$ are the parameters. Although $\{\phi_j\}$ are nonlinear functions, this approach is called the "linear model" by the representation, *e.g.* the support vector machine and the Gaussian process regression. The feedforward neural network can be regarded as an extension of this approach. For example, a feedforward neural network constructed by a hidden layer (two activation layers) is represented as

$$y_i = h\left(\sum_{k=1} w_{ik}^{(2)} h_k\left(\sum_{j=1} w_{kj}^{(1)} x_j\right)\right), \tag{3.2}$$

where h_k and h are functions called the nonlinear "activation functions" and $\{w_{ij}^{(1,2)}\}$ are the parameters called "weights". Especially, if the dummy variable equal to one is introduced, a weight corresponding to the variable is called the "bias" or the additive bias. An example for the activation function is the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{3.3}$$

Although it is difficult to represent the manipulation by the formula, the number of layers are able to be increased. The activation functions are defined to be differentiable. In other words, the derivative of the activation functions is able to be represented by some analytical functions. Another famous activation function is the rectified linear units function, called "ReLU," which is given by

$$\mathrm{ReLU}(x) = \begin{cases} 0, & x \le 0, \\ x, & x > 0. \end{cases} \tag{3.4}$$

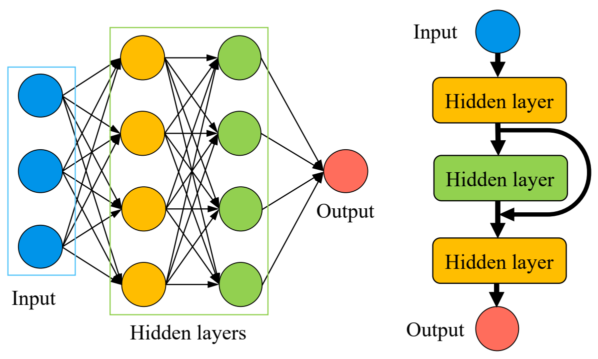


Figure 3.1: A schematic picture of neural networks for potential functions. The left figure shows a structure of the simplest feedforward neural networks, which we call “dense neural network”. The right figure represents the residual neural networks.

From its shape, the function is also called the ramp function. It is known that the ReLU function avoids the vanishing gradient of the neural nets because of the finite derivative. Although they are widely used and make learning successful, the ReLU function’s derivative at zero is not smooth. One alternative approach is using the sigmoid-weighted linear units function, called “SiLU” [110]. The function is defined by

$$\mathrm{SiLU}(x) = \frac{x}{1+e^{-x}}. \tag{3.5}$$

It is easily confirmed that the SiLU function is approximately a linear function or constant function far from zero. The function is smooth and a differentiable function. Another smooth function related to the ReLU function is the softplus function defined by

$$\mathrm{Softplus}(x) = \log(1+e^{x}). \tag{3.6}$$

The design of the neural networks affects the efficiency of the update (or optimization) of weights. To describe the architecture of the model, the graph expression is often used. The structure is called “feed-forward” because there is no cyclic path in the graph structure. Due to the complex relation between Eq. (3.1) and Eq. (3.2), the expressivity of the representation is unclear at first glance. However, the approximation power of the method is well-studied, and there are several theorems called the “universal approximation theorems” [6, 7, 9]. On the other hand, the reason why neural networks exhibit the generalization performance is still unclear. Some researchers indicate that the multi-layered structure causes the implicit regularization in simple cases far from the practical setup [111–113].

There are several types in the feedforward neural networks. The neural network discussed above is constructed by the fully-connected layers. In this thesis, we call such a feedforward neural network the “dense neural network.” One of the famous variants of feedforward neural networks is the “residual neural network,” whose schematic picture is shown in Fig. 3.1. The residual neural network is developed to avoid the vanishing gradients [114]. For high expressive power, neural networks likely have many hidden layers. Although they have the expressive power, acting the many nonlinear activation functions on the input occurs the vanishing gradients. The vanishing gradients spoil the learning, and the optimization of the parameters stops before the learning finishes. The essential point is acting the many nonlinear activation functions through the hidden layers. Thus, the residual neural network has the operation that skips the hidden layers:

$$y_i = x_i + h(x_i), \tag{3.7}$$

where the first term in the r.h.s is unactivated variables.

Once the network structure is determined, one has to decide the way to update the parameters. In general, one chooses the loss function $L(w)$, which represents the requirement for the neural network. If the neural network does not have the desirable properties, the loss function exhibits higher values. The loss function reaches the minimum value when the neural networks obtained the desirable properties. If the task is a regression problem, one of the famous loss functions is the sum of squared residuals called "mean squared error (MSE)" loss:

$$L(w) = \frac{1}{N}\sum_{i=1}^{N} \left\langle |y_i^* - y_i|^2 \right\rangle_D, \tag{3.8}$$

where y_i is the prediction, y_i^* is the training data which is prepared for learning, and $\langle\cdot\rangle_D$ is the average of the training data defined by

$$\langle g\rangle_D := \frac{1}{N_D}\sum_{n=1}^{N_D} g(x^{(n)}), \tag{3.9}$$

where N_D is the number of training data, and $x^{(n)}$ is the data point. The loss function is well-known by the least square methods. The loss function requires the neural networks to predict the value of the target function accurately. In this sense, the desirable property here is the approximation of the target function. There are various loss functions for this purpose. For instance, if the purpose is the classification problem, the loss function is likely the cross-entropy error function:

$$L(w) = -\sum_{i=1}^{N} \left\langle \left\{ y_i^* \log y_i + (1 - y_i^*) \log(1 - y_i) \right\} \right\rangle_D. \tag{3.10}$$

where y_i takes the value from zero to one. We note that we denote the loss function by omitting the variables as $L(w) = L$ if the representation is not confused.

The way to find the appropriate parameters is equivalent to the way to find the parameters achieving the minimum of the loss function. There are several ways to decrease the value of the loss function. These methods are often called the "optimizer," since the methods optimize the weights for the task. The most basic method is the gradient descent method, also known as the steepest descent method. The gradient descent method updates the weights by the gradient of the loss:

$$w \leftarrow w - \alpha \nabla L, \tag{3.11}$$

where α is the learning rate. If w reaches the extremum of the loss function, the optimization finishes. In this sense, the point w reached finally is not always the global minimum of the loss function. Contrary to the simple regression problems, the loss function of neural networks has non-convexity. The naive update method shows a poor performance due to the landscape of the loss function. Furthermore, because the training data set is likely huge, the calculation of the gradient becomes quite expensive. Thus, other update methods are used, and new approaches are proposed actively.

One of the most famous optimizers is the stochastic gradient descent method. In this method, the training data is divided into some groups called "mini-batch". The updates of the weights are performed by using the mini-batches. For instance, if one uses the MSE as the loss function, the weights are updated by

$$w \leftarrow w - \alpha \nabla L_{\mathrm{MB}}(w),\ L_{\mathrm{MB}}(w) := \frac{1}{N} \left\langle |y_i^* - y_i|^2 \right\rangle_{D_{\mathrm{MB}}}, \tag{3.12}$$

where $\langle\cdot\rangle_{D_{\mathrm{MB}}}$ is the average of mini-batch training data defined by

$$\langle g\rangle_{D_{\mathrm{MB}}} := \frac{1}{N_{D_{\mathrm{MB}}}}\sum_{n=1}^{N_{D_{\mathrm{MB}}}} g(x^{(n)}), \tag{3.13}$$

where $N_{D_{\mathrm{MB}}}$ is the size of the mini-batch training data, and $x^{(n)}$ is the data point. Usually, the mini-batches are reconstructed when all of the data is used. The reconstruction is realized by sampling from the training data. Thus, the combinations of the mini-batches are stochastically determined. The steepest descent updates the weights deterministically due to using the whole training data at once. It means that the steepest descent trajectory does not fluctuate, and it is hard to exit the local minimum. In this sense, the stochastic gradient descent method avoids being stuck and is expected to reach a lower point.

Although the stochastic gradient descent makes learning more successful than the steepest descent, there remain some problems. One of the largest issues is the constant learning rate. If the learning rate can be adaptively changed according to the loss function landscape, the updates are expected to become faster. There are several approaches [109]. The most famous method is the adaptive moment estimation method called the "Adam" optimizer [115]. In this method, the learning rate is adjusted by the local variables with the memory of updates:

$$\begin{aligned} m_i &\leftarrow \beta_1 m_i + (1-\beta_1)\partial_{\omega_i} L_{\mathrm{MB}}, \\ v_i &\leftarrow \beta_2 v_i + (1-\beta_2)\left(\partial_{\omega_i} L_{\mathrm{MB}}\right)^2, \\ \hat{m}_i &= \frac{m}{1-(\beta_1)^{N_{\mathrm{step}}}},\ \hat{v}_i = \frac{v_i}{1-(\beta_2)^{N_{\mathrm{step}}}}, \\ w_i &\leftarrow w_i - \alpha\frac{\hat{m}_i}{\sqrt{\hat{v}_i}+\epsilon}, \end{aligned} \tag{3.14}$$

where m and v are the local variables in the optimizer, β_1 and β_2 are the hyper parameters, N_{step} is the number of steps at the update, and ϵ is a small constant for avoiding zero-division error. In the Adam optimizer, m and v play a role of the adapted learning rate and their values contain the information of the past updates. From the first equation, the update of w is not just the value of the gradient $\nabla L_n(w)$, but added by the "momentum" of the updates. Thus, the updates are likely to keep the strength of learning. From the second equation, v suppresses the excess of the updates and changes the learning rate concerning the elements of the weights. If some of the weights are well updates but the others are not, the method adapts the learning rate to make the update small on the well-learned weights and *vice versa*. Thus, the weights are uniformly updated, and the update becomes more efficient than the standard gradient descent.

In the update process, the calculation to obtain the gradient of the loss function needs more consideration. The issue is that the derivative of the loss function by the layers near the input is harder than the derivative by the final layers. According to the chain rule of the differential, the derivative of the weighs needs a large number of matrix-vector products with increasing the layers of the neural network. By planning to calculate the derivatives properly, however, the calculation cost can be reducible. The algorithm is called "auto differentiation" and the manipulation is called "backpropagation" [14]. We suppose that the neural network is defined by Eq. (3.2) and the MSE is chosen as the loss function. For connivance, all of the activation function in each layer is the same type such as sigmoid function. Then, the derivative of the loss function by $w_{ij}^{(2)}$ is repressed by the chain rule as

$$\begin{aligned} \frac{\partial L(w)}{\partial w_{ik}^{(2)}} &= \sum_{i=1} \frac{\partial L(w)}{\partial a_i^{(2)}}\frac{\partial a_i^{(2)}}{\partial w_{ik}^{(2)}} \\ &= \sum_{i=1} \frac{\partial L(w)}{\partial a_i^{(2)}} x_k^{(1)}, \\ \frac{\partial L(w)}{\partial a_i^{(2)}} &= \frac{\partial L(w)}{\partial y_i}\frac{\partial y_i}{\partial a_i^{(2)}} \\ &= \frac{\partial L(w)}{\partial y_i} h'\left(a_i^{(2)}\right), \\ \frac{\partial L(w)}{\partial y_i} &= \frac{2}{N_D}(y_i^* - y_i), \end{aligned} \tag{3.15}$$

where h' is the derivative of the activation function h, and $a_k^{(1)}, a_k^{(2)}, x_k^{(1)}, y_i$ are defined as

$$\begin{aligned} a_k^{(1)} &= \sum_{j=1} w_{kj}^{(1)} x_j, \quad x_k^{(1)} = h\left(a_k^{(1)}\right), \\ a_i^{(2)} &= \sum_{k=1} w_{ik}^{(2)} x_k^{(1)}, \quad y_i = h\left(a_i^{(2)}\right). \end{aligned} \tag{3.16}$$

Although the derivation of the chain rule starts from the first equation in Eq. (3.15), it easy to obtain the derivative of the loss function in each by starting from the final equation. We call the calculation of each layer started from the input the "forward propagation." For the automatic differentiation, the output at each layer is stored through the forward propagation. It is noted that each derivative of the loss function is constructed just the differentiation by the output at the layer. Thus, if the process to obtain the derivative of the loss function by each output is started from the final layer, we can calculate all derivatives at once. Furthermore, we define the activation function so that its derivative is easily calculated. Hence, it is not necessary to calculate the derivative of the activation function by numerical differentiation. This way, we can obtain all the derivatives automatically just through the backpropagation.

3.2 Normalizing Flow

Here, we introduce the particular structure of the neural networks to represent the canonical transformation. It is called the "Normalizing flow" [80]. This neural network is developed for the generative model. The generative model's task is to learn the distribution of the training data and quickly produce the same distribution. It learns the transformation from the distribution that can be sampled easily to the correct distribution. If the neural network producing the distribution is obtained, one can generate other samples obeying the same distribution with the training data. There are other approaches to construct the generative model [4]. One of the most famous approaches is the generative adversarial network, also known as GAN [116]. This method is constructed by the two neural networks playing the roles of generator and discriminator. The two neural networks deceive each other and improve the generated samples to be indistinguishable from the real. One of the disadvantages of such methods is the difficulty of the process of reproducing the data. For normalizing flow, the structures make the reproducing process easy.

The normalizing flow is also constructed with the stack of layers. The difference between the normalizing flow and the standard feedforward neural network is the structure of the layers. For normalizing flow, the layer is made to be invertible and bijective. This structure makes the inverse transformation easy. Let x be the domain and z be the codomain. If we denote each layer of the normalizing flow as f_i, the whole transformation is written as

$$z = f_n \circ f_{n-1} \circ \cdots \circ f_1(x), \tag{3.17}$$

where $\{f_i\}$ are defined as invertible and bijective functions. By construction, the whole transformation becomes a bijective function inevitably. There are several types of normalizing flow characterized by the choice of $\{f_i\}$. For instance, the function is constructed as the real-valued non-volume preserving transformation, which is called the RealNVP model [117], The essential point is that the domain is divided into two parts, and the nonlinear function transforms only a part:

$$z_< = x_<, \tag{3.18}$$

$$z_> = x_> \odot \exp(s(x_<)) + t(x_<), \tag{3.19}$$

where $x_<$ and $x_>$ denote the divided domains, and s and t are the nonlinear functions playing a role of scaling and translation, and $\odot$ stands for the Hadamard product. It is noted that the exponential function in the second equation is not the matrix function but acts on each element. In this sense, the realNVP model is an affine transformation represented by the feedforward neural networks s and t. Therefore, the transformation layer is also called the “affine coupling layer” [117, 118]. There are several separations of the data into two parts. In this thesis, we apply the staggered mask. For instance, if the input dimension is 8, we separate the variables as $x_< = x_0, x_2, \dots, x_6$ and $x_> = x_1, x_3, \dots, x_7$. Furthermore, we act the coupling layer on two parts equally by flipping the mask. In other words, we avoid acting the nonlinear transformation on just a part of variables, but on all the variables by flipping the mask. We note that all of the coupling layers are different and have their parameters.

It is easily confirmed that the layer is bijective and invertible through the Jacobian. The Jacobian of the transformation is

$$\frac{\partial z}{\partial x} = \begin{pmatrix} I & 0 \\ \frac{\partial z_>}{\partial x_<} & \mathrm{diag}\left(e^{s(x_>)}\right) \end{pmatrix}. \tag{3.20}$$

Obviously, the determinant of the Jacobian is

$$\det\left(\frac{\partial z}{\partial x}\right) = \exp\left(\sum s(x_>)\right) \neq 0. \tag{3.21}$$

It means that the Jacobian is a regular matrix. Furthermore, the model has specific features that the inverse of the transformation is computed as simply as the forward propagation. In fact, the inverse is explicitly given by

$$\begin{aligned} x_< &= z_<, \\ x_> &= \exp(-s(z_<)) \odot (z_> - t(z_<)) . \end{aligned} \tag{3.22}$$

This calculation is no more complex than the forward process. It is reported that the affine-coupling normalizing flow has a property of universal approximation under certain conditions [119]. We use the dense neural networks as s and t in Eq. (3.19).

3.3 Neural Ordinary Differential Equation

In this section, we introduce the methods to deal with the differentiation of time development. For the calculation of the gradient of time development, we apply the adjoint method to neural networks [81]. The approach is known as the neural ordinary differential equation (NODE). The adjoint method is an approach to calculate the gradient by the integration in the adjoint space [82]. There are a few ways to introduce the method. Here, we use the correspondence of the numerical integration scheme and the residual neural networks defined by Eq. (3.7). The structure of the residual neural networks can be regarded as a difference equation. From the view of the correspondence, we can come up with the “continuous” residual neural networks, which is the so-called neural ordinary differential equation as

$$x = x + h(x)\Delta t, \tag{3.23}$$

$$\frac{dx}{dt} = h(x). \tag{3.24}$$

The residual neural networks are regarded as one of the differentiations of ordinary differential equations through the interpretation. Thus, we can use other numerical integration methods, *e.g.* the Runge-Kutta method. Although the generalization provides the prescription to construct the “infinite” number of the hidden layers, the backpropagation has the difficulty. If the number of updates increases, the memory cost of the calculation graph becomes proportionally large. To reduce the memory cost, the adjoint method developed in the field of optimal control is useful [82]. The adjoint method is the differential equation, which obeys the derivative of the scalar function of time-dependent variables:

$$\begin{aligned}\frac{dx}{dt} &= f(x,\theta),\\ \frac{d\lambda}{dt} &= -\left(\frac{\partial f(x,\theta)}{\partial x}\right)^T \lambda,\\ \frac{d\lambda^\theta}{dt} &= -\left(\frac{\partial f(x,\theta)}{\partial \theta}\right)^T \lambda^\theta,\end{aligned} \tag{3.25}$$

where λ is the derivative of the scalar function of $x(t)$ and λ^θ is the one for θ. Needless to say, in the Hamiltonian mechanics, x corresponds to (q,p) and f is $(\partial_p H, -\partial_q H)$. The adjoint method can be regarded as the continuous backpropagation of NODE via the following derivation (see Appendix in [81] or Eq. (A.20)). The method requires only the final state of $x(t)$, and we can calculate the derivation of the loss function by integrating the adjoint equation. In the original paper, the Runge-Kutta methods are used for the numerical integration method. However, the symplectic integrators also can be adopted for the adjoint method. The detail is discussed in Appendix. A.

3.4 Applications to Hamiltonian Systems

3.4.1 Neural canonical transformation

From the point of view of the normalizing flow, the canonical transformation can be represented by the neural networks [78, 79]. Here, we call the canonical transformation represented by the neural network "neural canonical transformation" according to [78]. Several approaches use the neural canonical transformations. The approaches are characterized by the construction of neural canonical transformations and the purpose of the loss function design.

In [78], they use the realNVP for the point transformation acting on the position. According to the generating function of Eq. (2.17), the canonical transformation is

$$\begin{aligned}Q_i &= F_i(q),\\ P_i &= \sum_{j=1}^{N} p_j \frac{\partial q_j}{\partial Q_i},\end{aligned} \tag{3.26}$$

where F_i is the realNVP model. At first glance, the r.h.s. in the second equation seems to be difficult to calculate. By the automatic differentiation and the feature of normalizing flows, however, the quantities are obtained as

$$\begin{aligned}q_i &= F_i^{-1}(Q),\\ P &= \nabla_Q(p\cdot q).\end{aligned} \tag{3.27}$$

The first operation can be performed by the inverse of the normalizing flow. The computation graph is created through the operation, and the derivative of the product $p \cdot q$ is produced by the automatic differentiation mechanism.

Their purpose is to learn the relevant coordinates of the system. Thus, they attempt to find the canonical transformation that maps the system to interdependent harmonic oscillators approximately. They understand the feature of the motion by the normal modes obtained by the neural networks. For the aim, they use the likelihood approach to make the loss function. Let $K(z)$ be the Hamiltonian of the interdependent harmonic oscillators

$$K(z) = \sum_{k=1}^{N} \frac{P_k^2 + \omega_k^2 Q_k^2}{2}, \tag{3.28}$$

where ω_k is the coefficients to be learned. ω_k represents the feature of the dynamics of the learned distribution. The neural network learns the canonical transformation that maps the training data distribution to the normal distribution characterized by the Hamiltonian Eq. (3.28). They use the variational free energy and negative log-likelihood derived from the Kullback–Leibler divergence [120]. The Kullback-Leibler divergence is defined by

$$\mathrm{KL}(\rho|\pi) = \int dx \rho \ln\left(\frac{\rho}{\pi}\right), \tag{3.29}$$

where ρ is the Boltzmann distribution of Eq. (3.28) and π is the empirical distribution or *vice versa*. The quantity measures the distance between the two distributions.

In [79], they represent the canonical transformation by several popular invertible layers modified to be symplectic. They use mainly three blocks of the transformation, the symplectic additive coupling, the symplectic linear, and the zero center layer. First, the symplectic additive coupling layer is introduced as

$$(Q, P) = (q, p + NN(q)), \ (q, p) = (Q, P - NN(Q)), \tag{3.30}$$

where NN is an neural network satisfying the irrotational condition:

$$\frac{\partial NN_j}{\partial q_i} = \frac{\partial NN_i}{\partial q_j}. \tag{3.31}$$

They realize the neural network by restricting the weights of the layers in NN. The second layer is the symplectic linear. It is represented by the pre-Iwasawa decomposition Eq. (2.21). To parameterization of the symplectic matrix, K in $U(n)$ is represented by the Householder transformation [79, 121]. The Householder transformation is represented as

$$R_v = I - 2\frac{vv^\dagger}{||v||^2}, \tag{3.32}$$

where v is the reflection vector. Thus, K in $U(n)$ is represented as

$$K = \prod_{i=1}^{N} R_{v_i} \mathrm{diag}(e^{i\phi}), \tag{3.33}$$

where ϕ is a $2N$ vector. The final layer is the zero center layer. It is a part of the batch normalization introduced previously. Their purpose is to learn the symmetry of the integrable systems. They define the learning of the symmetry as learning of the action-angle variables. Thus, the desired canonical transformation is a map from the real coordinates to the action-angle variables. According to the diffeomorphism of the integrable system, there are the coordinates that describe the action-angle variables as

$$I_k = \frac{P_k^2 + Q_k^2}{2}, \quad \theta_k = \tan^{-1}\left(\frac{P_k}{Q_k}\right). \tag{3.34}$$

They make the loss function, which implies the conservation of the action variables. The training data is the trajectory generated by integrating the equation of motion. Let N_{time} be the number of the time points in the trajectory for learning. Eventually, the loss function is defined as

$$L = \frac{1}{NN_{\mathrm{time}}} \sum_{n=1,k=1} \left\langle |I_k(t_n) - I_k(t_{n-1})|^2 \right\rangle_D . \tag{3.35}$$

The data set is sampled from the Gaussian distribution. According to the loss function, this approach can be regarded as unsupervised learning. They learn the several integrable systems but not the Toda lattice. As mentioned below, we learn the canonical transformation to obtain the action-angle variables of the Toda lattice.

3.4.2 Learning dynamics of systems

In this section, we introduce some applications to learn the dynamics of physics by the neural networks. The approaches mentioned below include some structures based on the law of physics. Such approaches are called “physics-informed” or “physics-enhanced” neural networks and attract attention recently. In one of the seminal articles applying the neural networks to the Hamiltonian mechanics, the Hamiltonian is represented by the neural network, which is called the “Hamiltonian Neural Network” [122]. The purpose is to learn the dynamics from the time series data and predict the time series in the future accurately. They represent the Hamiltonian explicitly by the neural network and calculate the force from the Hamiltonian by the auto differentiation. The loss function is also characteristic. We denote the Hamiltonian represented by the neural network as H_θ. The training data is the trajectories of the system, which is described by the true Hamiltonian. Then, the loss function is defined as

$$L = \frac{1}{NN_{\text{time}}} \sum_{n=1,i=1} \left\langle \left| \frac{\partial H_\theta}{\partial p_i} - \frac{dq_i(t_n)}{dt} \right|^2 + \left| \frac{\partial H_\theta}{\partial q_i} + \frac{dp_i(t_n)}{dt} \right|^2 \right\rangle_D . \tag{3.36}$$

The derivative of the Hamiltonian by the canonical coordinates is carried out by the auto differentiation. Since the neural network is designed to have the properties of the Hamiltonian mechanics, the prediction of the model conserves the energy defined by the neural network. Due to these properties, the prediction is more accurate than the model that predicts the force instead of the Hamiltonian. This work pointed out that designing neural networks with physical insights is more efficient in parameterizing the systems. The Hamiltonian neural networks learn the models exhibiting Hamiltonian chaos and order [123]. The model can predict even the chaos trajectory by the learned Hamiltonian.

The subsequential approaches are being reported. The Lagrangian neural network is the model representing the Lagrangian instead of the Hamiltonian [124]. In this approach, they compute the acceleration of the coordinates by the Lagrangian. The loss function is defined in terms of the discrepancy between the prediction and the training data. The neural network can learn the system described by the Lagrangian formalism but not the Hamiltonian formalism. In the above two articles, they do not use the integration of the parameterized classical mechanics to evaluate the loss function. For instance, in the Hamiltonian neural network, they evaluate the parameters by the force at the time point. It is natural to come up with the loss function, including the discrepancy between the trajectories generated by the models. There are several approaches reported, but some of them do not use the adjoint method, but the ordinary backpropagation [125, 126]. They do not use the adjoint method because the integration method is time-consuming as long as the size of the model is relatively small. In the ordinary backpropagation, the derivatives of the hidden layers are stored to reduce the time to evaluate the function. In contrast, the adjoint method reduces the memory cost of the updates at the cost of the evaluation. In one of the methods with the adjoint method, they transform the coordinates to integrate them with symplectic properties [127]. In the method, the model also learns the dynamics of the external variables representing the control acting on the system. As mentioned below, in contrast to our method, they use the 4th-order Runge-Kutta method as the integrator.

So far, we focus on the articles that seek the Hamiltonian or the differential equation of the systems. However, learning dynamical systems is not restricted to the equations but also the systems’ characteristics. One of the examples is to learn the physical concepts of the systems automatically [33]. They use a neural network similar to the variational autoencoder neural network. The variational autoencoder has the characteristic learning called the “representation learning” or “feature learning” [128] The model learns the reduced information of the training data and relevant lower-dimensional space of the system. The process is similar to the thinking of the physicists. The physical law is the relevant description of the system with a few variables. They expect the neural network to imitate the activities and find the concepts automatically. In several examples, including heliocentric solar systems, the neural network shows the ability to find the law of motions.

Chapter 4

Exploration of Classical Integrable Systems Assisted by Neural Networks

4.1 Preliminary

In this section, we introduce our purpose and a brief overview of the discussion below.

Our purpose is to propose a method of helping us to explore the classical integrable systems. Furthermore, the construction of the integrable system is different from the previous approaches, *e.g.* an ansatz for the Lax pair. We expect that unrevealed integrable systems are found by the proposed method, and the obtained systems indicate some intriguing physics. Moreover, we attempt to establish the procedure to explore the systems assisted by neural networks. It is still a problem to validate the machine learning results in the cases corresponding to the third direction of the applications introduced in Chap. 1. Through the exploration of the classical integrable system, we look for an appropriate protocol for such applications.

For the construction of classical integrable systems, there are several approaches we can come up with whether the approaches are possible or not. To begin with, we consider the possible constructions of the integrable system from the viewpoint of the transformations in Sect. 4.2.1. After that, we adopt the approach for constructing the systems from the transformed Hamiltonian, which is described by the action variables. The construction is hard to be achieved except for the simplest systems. It is noted that the approach assumes the motions of the integrable system are bounded for defining the action-angle variables. In Sect. 4.2.2, we introduce the structure of the neural networks representing the canonical transformation and the potential function. Since there are still some realization of the approach, especially the loss function, we compare some types of the error functions with two tasks, which are the parts of our desirable task in Sect. 4.2.4 and Sect. 4.2.5 After the evaluation, we construct the whole method and test it in Sect. 4.2.6 and Sect. 4.2.7.

In particular, we use the Toda lattice to operate all of the tasks and tests. There are three reasons. The first reason is that the Toda lattice has a complex potential function. The potential function of the Toda lattice is an exponential function. In harmonic oscillators, the force is described by a linear function, which is too simple for testing. The second reason is that the construction of the action variables is already known in the Toda lattice. Therefore, we can test whether the neural network learns the canonical transformation or not. Moreover, the transformation is more complicated than harmonic oscillators. It means that we can examine the representation power of the neural networks by the system. The third reason is that the thermodynamic properties of the Toda lattice are well-known. As mentioned below, we make the training data sampled from the Boltzmann distribution. Therefore, we can validate the data from the viewpoint of the thermodynamics of the Toda lattice.

Finally, in Sect. 4.3, we attempt to explore unrevealed integrable systems by the proposed method. The trained neural networks suggest the potential function which is the integrable system characterized by a reduced Hamiltonian with perturbations. We extract the insights into the candidates of the integrable systems from the behavior of the neural networks. As a result, we determine the analytical representation

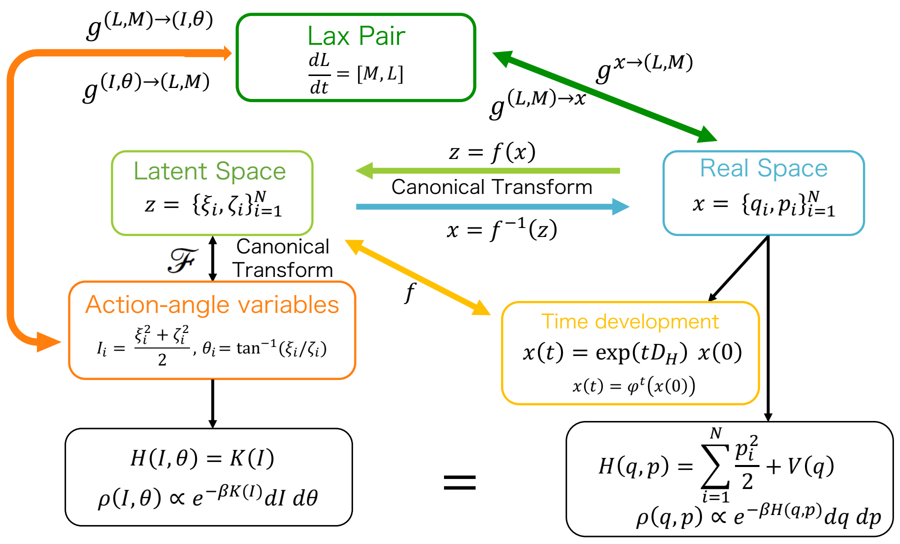

$g^{(L,M)\to(I,\theta)}$
$g^{(I,\theta)\to(L,M)}$
Lax Pair
$\frac{dL}{dt} = [M, L]$
$g^{x\to(L,M)}$
$g^{(L,M)\to x}$
Latent Space
$z = \{\xi_i, \zeta_i\}_{i=1}^N$
$z = f(x)$
Canonical Transform
$x = f^{-1}(z)$
Real Space
$x = \{q_i, p_i\}_{i=1}^N$
$\mathscr{F}$
Canonical Transform
f
Action-angle variables
$I_i = \frac{\xi_i^2 + \zeta_i^2}{2}, \theta_i = \tan^{-1}(\xi_i/\zeta_i)$
Time development
$x(t) = \exp(tD_H)\ x(0)$
$x(t) = \varphi^t(x(0))$
$H(I,\theta) = K(I)$
$\rho(I,\theta) \propto e^{-\beta K(I)} dI\, d\theta$
=
$H(q,p) = \sum_{i=1}^N \frac{p_i^2}{2} + V(q)$
$\rho(q,p) \propto e^{-\beta H(q,p)} dq\, dp$

Figure 4.1: Diagrams of the whole map related to obtain the action-angle variables of the integrable systems.

of the suggested integrable system.

4.2 Methodology

4.2.1 Strategy

Our purpose is to explore new integrable systems. We consider a strategy to achieve the purpose by focusing on the representation power of the neural networks. As mentioned in Chap. 3, there are several abilities in the neural networks. Thus, we can come up with several strategies to explore integrable systems. To devise a strategy, we see the transformations, functions, and coordinates involved with procedures to obtain the conserved quantities. Especially, we focus on the process to obtain the action-angle variables, which is a manifestation of integrable systems. Hereafter, we consider N particles systems. In Fig. 4.1, we present the whole transformations to obtain the action-angle variables. There are four coordinates, seven transformations, and four functions.

Coordinates $z, x, \{I_i, \theta_i\}_{i=1}^N, (L, M)$

Transformations $f, \mathscr{F}, \phi^t, g^{x\to(L,M)}, g^{(L,M)\to x}, g^{(L,M)\to(I,\theta)}, g^{(I,\theta)\to(L,M)}$

Functions $V(x), K(I)$

First of all, we see spaces. x and z are canonical coordinates. x stands for the pair of the trivial canonical coordinates denoted as $\{q_i, p_i\}_{i=1}^N$. x is nothing but the coordinates defined naturally. In this thesis, we call the space the "real space". The variables are observed in reality. $z = \{\xi_i, \zeta_i\}_{i=1}^N$ are the canonical coordinates whose radii and angles are related to the action-angle variables. In other words, the coordinates satisfy

$$\begin{aligned} I_i &= \frac{\xi_i^2 + \zeta_i^2}{2}, \\ \theta_i &= \tan^{-1}\left(\frac{\xi_i}{\zeta_i}\right) \end{aligned} \tag{4.1}$$

where $\{I_i, \theta_i\}_{i=1}^N$ are the action-angle variables introduced in Sect. 2.2.1. They play a role of the generalized momenta and positions, respectively. These coordinates are introduced to treat the canonical transformation by the neural networks. Here, the space described by these coordinates is called the "latent space". The existence of such space is proved in some cases [99, 100, 129–131]. According with the success of the previous approach [79], we use the latent space to represent the action-angle variables. At least, the existence of the action-angle variables is the manifestation of integrability. Thus, if the canonical transformation between the real space to the action-angle variables is found, the system is revealed to be an integrable system. (L, M) is the Lax pair. As we see the geometrical aspects of the Lax pair in Sect. 2.2.4, they are related to the Lie groups. In this sense, the Lax pair is regarded as the equation of motion on the space whose elements are defined by matrices.

Next, we focus on the transformations. f and $\mathscr{F}$ are canonical transformations. It is noted that the canonical transformations are bijective and satisfy the condition Eq. (2.9). f is a canonical transformation that maps the real space to the latent space. $\mathscr{F}$ is the transformation between the latent space and the action-angle variables. The transformation is equivalent to the canonical transformation to the action-angle variables in the harmonic oscillator chain defined by Eq. (2.33). Since the composition of f and $\mathscr{F}$ is a canonical transformation, we find the canonical transformation from real space to the action-angle variables if we find f. ϕ^t is the time-development operator defined by Eq. (2.7). The operator is the exponential map by the Hamiltonian. Thus, once the Hamiltonian is defined, we obtain the map ϕ^t according to Eq. (2.7). $g^{x\to(L,M)}$ is a transformation from the real coordinates to the Lax pair. For instance, the process of constructing the Lax pair from the Toda lattice corresponds to the transformation. The concrete expression of the matrix is shown in Appendix. B.2.2. It is noted that the transformation is not determined uniquely. In fact, there are a few different forms of the Lax pair for the Toda lattice [93, 96, 97]. $g^{(L,M)\to x}$ is a transformation from the Lax pair to the real coordinates. It is obvious that one can obtain $g^{(L,M)\to x}$ as the inverse transformation if $g^{x\to(L,M)}$ is given. $g^{(L,M)\to(I,\theta)}$ is a transformation form the Lax pair to the action-angle variables. An example is constructing the action-angle variables for the Toda lattice, see Eq. (2.43). It is noted that a composition of $g^{x\to(L,M)}$ and $g^{(L,M)\to(I,\theta)}$ is a canonical transformation. Thus, if one finds the transformation $g^{(L,M)\to(I,\theta)} \circ g^{x\to(L,M)}$, the system is integrable. $g^{(I,\theta)\to(L,M)}$ is the transformation form the action-angle variables to the Lax pair. If one obtains the action-angle variables, there is a trivial Lax pair from the prescription Eq. (2.41).

Finally, we see the functions. There are two functions, $V(q)$ and $K(I)$. $V(q)$ is a potential function whose domain is the real space. The Hamiltonian defined by the kinetic term and potential function is called the "natural Hamiltonian" [76]. $K(I)$ is the Hamiltonian of the action-angle variables. It is noted that K depends on only the action variables.

Before considering strategies, we define "discovery" in the exploration of integrable systems. In other words, we consider the criterion to find an integrable model. There are several criteria to define the discovery of an integrable system. In an extreme case, one can argue that an integrable system is found by just making Hamiltonian depending on only action variables. Our interest is, however, about the properties of the potential function $V(q)$ and phenomena in the real space. The potential function defined in the real space is regarded as the realization in reality. Thus, it is not enough to insist that one finds an integrable system by just making $K(I)$. In this thesis, we call it "discovery" to find the potential function $V(q)$, whose Hamiltonian can be transformed to a transformed Hamiltonian $K(I)$.

From the observation, one can come up with several strategies to explore integrable systems. The strategies are found from the diagram of transformations in Fig. 4.1. In principle, all components in Fig. 4.1 can be represented by neural network or other way. If one attempts to make a supervised learning method, one uses the true data of two components in the diagram. If we regard the components in Fig. 4.1 as nodes in graph theory, two components are a tail and a head of a path in the graph. In another way, if one attempts to make an unsupervised learning method, one creates a closed-loop in the graph. This way, we can make several strategies to find integrable systems by neural networks.

Here, we devise a strategy from the definition of integrable systems directly. We represent the potential function $V(q)$ and the canonical transformation f by the neural networks. Furthermore, we make the loss function in terms of the action-angle variables. According to the definition of integrable systems, a system must have the special canonical coordinates, the action-angle variables. If the requirement has a mathematical representation, we can formulate the loss function that meets the minimum requirement when the requirement is satisfied. From this point of view, we represent the definition of the action-angle variables as the loss function. It is easy to come up with the following loss functions:

$$L = \frac{1}{NN_{\text{time}}} \sum_{i=1}^{N} \sum_{k=1}^{N_{\text{time}}} \ell \left(I_i(t_0) - I_i(t_k) \right), \tag{4.2}$$

or

$$L = \frac{1}{N} \sum_{i=1}^{N} \ell \left(\partial_t I_i(x) \right), \tag{4.3}$$

where ℓ represents an error function. The error function includes the operation to get the average of the training data. One of the examples is the mean squared error function introduced in Sect. 3.1. Note that in the above loss functions, the canonical transformation $\mathscr{F}$ is omitted for simplicity. The first equation imposes that the action variables have to be equal to their initial values of the time series. The time series are generated by the true potential function or the potential function represented by neural networks. In the present thesis, we use the fourth-order symplectic integrator to generate the trajectories [132]. In the latter case, we use the adjoint method for reducing the memory costs of the calculation of gradients. In particular, we adopt the symplectic integrator with the adjoint method. The second equation requires the time derivatives to be zero. It is nothing but the definition of the conserved quantities. The loss function does indicate the requirements for the variables or the canonical transformations and those for the Hamiltonian. Through the time series or time derivative, the loss function affects the Hamiltonian. It means that the represented potential function $V(q)$ can learn with the canonical transformation to the action-angle variables simultaneously. This way, we can obtain the integrable system by using unsupervised learning.

We note that the loss function provides just the necessary condition. For instance, if we use an infinite number of the time points for Eq. (4.2), we can obtain the perfect action variables. However, such a situation can not be realized due to the limit of numerical resources. Similarly, there are restrictions in Eq. (4.3). If we evaluate the loss function at all phase space points, it gives us the perfect action variables. However, practically, there are limitations to the memory and performance of computers. Thus, it is hopeless to make the loss function that provides the necessary and sufficient conditions. The fact indicates that the system we obtain by the neural network approach is a perturbed integrable system. Although the loss functions are difficult to present the true integrable systems, they are expected to give us powerful insights.

Although both loss functions require the existence of the action-angle variables, the second loss function is more complicated. As mentioned above, the second loss function requires enough data to fill the whole phase space, including the trajectories generated by the time development. Thus, even if we train the neural networks in the input data adequately, the potential function and the canonical transformation do not always exhibit the conservation of the action variables. Therefore, we use the second loss function represented in Eq. (4.2).

Here, we note the difference of our approach from the previous researches introduced in Sect. 3.4.1 and Sect. 3.4.2. The researches introduced in these sections are aimed to extract the transformations or the potential function from the data in the real space. Therefore, they use neural networks to analysis of the data. In our approach, we make the data in the latent space. In this sense, our method is completely different from the previous researches using neural networks to classical mechanics.

In order to use the loss function, we restrict the potential function $V(q)$. In this thesis, we consider a particular group of potential functions that depend on the difference between two positions $q_i - q_j$. If we restrict the potential functions to such a group, the motion of the center of mass is trivially conserved. It means that we can avoid the diffusion of the particles far away and restrict the domain of canonical transformations. In general, if the domain of the transformation becomes large, the number of layers in the normalizing flow increases. Thus, the elimination of the motion of the center of mass reduces the cost of constructing the neural networks. Furthermore, the manipulation also avoids the divergence of the learning action variables. The loss function evaluates the difference between the value of action variables at two time points. In the initial stage of the learning process, the difference can be huge. To avoid the divergence of the loss function as possible, we eliminate the central motion. This is the reason why we restrict the potential function to the specific group. Concretely, we suppose the potential function as the following three cases:

$$\begin{aligned} V_{\text{chain}}(q) &= \sum_{i=1}^{N} v(r_{ii+1}),\ r_{NN+1} = r_{N1} \\ V_{\text{fully}}(q) &= \sum_{i<j} v(r_{ij}), \\ V_{\text{multi-body}}(q) &= v(r_{12}, r_{13}, \ldots, r_{N-1N}), \end{aligned} \tag{4.4}$$

where v is a function represented by the neural networks, and r_{ij} represents the difference of the two positions $r_{ij} = q_i - q_j$. The first potential function is defined in the chain with periodic boundary conditions. The second one is a fully-connected potential function. The final one is the potential representing arbitrary translational invariant interactions. These potential functions conserve the center of motion and are appropriate for our aim.

Hereafter, for convenience, we call the canonical transformation represented by the neural network the "neural canonical transformation" and the potential function the "neural potential function". The name of the neural canonical transformation is according to the previous research [78].

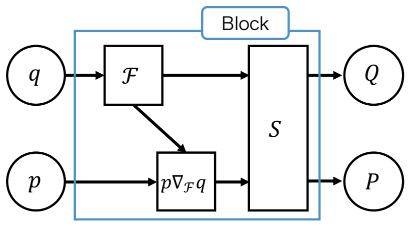
In the above discussion, it is mentioned that the loss function can train the neural canonical transformation and the neural potential function simultaneously. The approach, however, does not exclude the known systems explicitly. The neural networks are expected to find an integrable system close to the initial state. In this approach, the elements we can manipulate are the initial parameters and the structure of the neural networks. Thus, it is not easy to control the direction of the learning procedure. Furthermore, the approach does not exclude the systems whose physical parameters are different, but the potential is essentially equivalent. The shape of the potential determines the characteristic property of the dynamics. For instance, the Toda lattice has two parameters, α and J. In the simple approach, the neural networks find a Toda lattice but do not distinguish another Toda lattice whose parameters are different from the former. In the worst case, the neural networks find numberless systems essentially equivalent to Toda lattice. Therefore, we add the operation which directs the learning process to obtain unknown systems.

In this thesis, we assume the transformed Hamiltonian $K(I)$ and give the constraints derived from the assumption. $K(I)$ is the desired Hamiltonian itself. Thus, through the given $K(I)$, we specify the integrable system obtained by the neural networks. There are several ways to add the assumption to the loss function. We attempt to evaluate two approaches. One of them is a simpler approach: adding the loss function about the difference of the energies:

$$L = L_I + L_E,\ L_E := \ell(H - K), \tag{4.5}$$

where L_I is the loss function defined by Eq. (4.2) and L_E is the loss function about the energy. The loss function L_E requires the Hamiltonian to be the desirable system. The approach is straightforward and straightforward. If we know the Hamiltonian desired by the action variables, however, we can sample the action-angle variables by the Boltzmann distribution. From this viewpoint, the other method is come up with: adding the loss function about the thermodynamic properties, especially the Boltzmann distribution. According to the equivalence of the Hamiltonian $H(x)$ and $K(I)$ (see Fig. 4.1), if we make the training data from the distribution characterized by $K(I)$, the distribution $\rho(p)$ is equal to the Boltzmann distribution:

$$L = L_I + L_E + L_p, \tag{4.6}$$



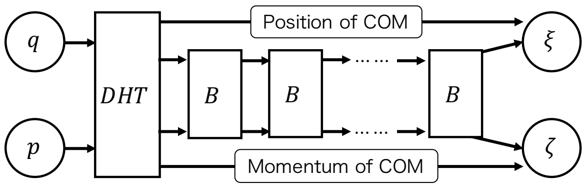


Figure 4.2: Diagrams of our canonical transformations from real space to the latent space. The left figure shows the block, which is a part of whole canonical transformation. The block is a combination of point transformation and symplectic linear transformation. $\mathcal{F}$ is a bijective map represented by RealNVP. The transformation of p is derived from the generator function defined by Eq. (2.17). S stands for the symplectic linear transformation. The right figure shows the whole structure of the canonical transformation from the real space to the latent space. DHT stands for the discrete Hartley transformation and B stands for the block. COM stands for the center of mass.

where L_p is the loss function of the momentum distribution. It is defined as

$$L_p := \frac{1}{N}\sum_{i=1}^{N}\left\{\ell(\langle p_i\rangle_D) + \ell(\langle p_i^2\rangle_D - \langle p_i^2\rangle_{\text{thermal}})\right\}, \tag{4.7}$$

where $\langle\cdot\rangle_{\text{thermal}}$ is an average over the Boltzmann distribution. In this approach, the distribution of the momentum p is obtained from the training data (I, θ) by the transformation $f^{-1} \circ \mathscr{F}^{-1}$. The approach provides us a way to create the training data and more information about the desirable Hamiltonian. In the simple approach defined by Eq. (4.5), the appropriate data set is unclear. Conversely, for the latter approach, the data set is produced from the thermodynamic perspective. Thus, we expect the approach defined by Eq. (4.6) is more appropriate than the other, and we adopt it.

Here, we explain the training data. In this thesis, we treat the two types of training data. The first type is sampled from a simple Gaussian distribution. The variables $x = q_i, p_i{}_{i=1}^{N}$ are sampled from the $2N$ Gaussian distribution. In this case, we use the first loss function defined by Eq. (4.5). The other type is sampled from the Boltzmann distribution of $K(I)$. Concretely, we sample the pair of action-angle variables $\{I_i, \theta_i\}_{i=1}^{N}$ and input the data to the neural canonical transformation as points in the latent space, $\{\xi_i, \zeta_i\}_{i=1}^{N}$. In this approach, we use the second loss function defined by Eq. (4.6).

Consequently, we represent the process to obtain the action-angle variables of integrable systems and neural potential function.

4.2.2 Structure of neural networks

We represent the canonical transformation and the potential function by neural networks.

First, we introduce the neural networks representing the potential functions. As mentioned above, we treat the some kinds of potential functions defined by Eq. (4.4). We use two kinds of neural networks, the dense neural networks and the residual neural networks, shown in Fig. 3.1. The dense neural network is one of the most popular structures. The residual neural network is used for the representation of the potential function in a previous research [133]. There are two parameters: the number of hidden layers and the dimension of the hidden layers. In general, each hidden layer can be different from the others. For instance, it is possible to use a different dimension of weights in each hidden layer. In this research, we use the same dimension for all the hidden layers for simplicity and convenience. The dimension of the hidden layers is chosen to be 128. On the other hand, the number of hidden layers is changed to obtain the appropriate performance. The dimension of the input is different depending on the type of potential function. For instance, the input dimension of the multi-body interaction potential function is $\frac{N(N-1)}{2}$. On the other hand, for the nearest-neighbor interaction, the dimension is just N.

Next, we explain the structure of the canonical transformation. We construct the canonical transformation f, the map from the real space to the latent space. The structure of our canonical transformation is shown in Fig. 4.2. The left figure of Fig. 4.2 shows the “canonical transformation block”. The canonical transformation is realized by a stack of blocks and the discrete Hartley transformation shown in the right figure of Fig. 4.2. The canonical transformation block consists of two canonical transformations. One of them is the canonical transformation generated by the point transformation, defined by Eq. (2.17). The point transformation is represented by RealNVP defined by Eq. (3.19). This realization of canonical transformation is applied in [78], which is described in Sect. 3.4.1. For the scale and the translation transformation in RealNVP, we use the dense neural network. The canonical neural transformation does not represent the transformation combining the momentum and position. The point transformation maps the position to a newly defined position. Thus, we apply the symplectic linear transformation, which is the other part of the block. The parametrization is mainly according to the method used in [79] but slightly different. We increase the number of vectors of the the Householder transformation in the symplectic linear transformation. In the previous research, they represent the transformation by a reflection vector so that the numerical cost is reduced. Through some trial experiments, we find that it is difficult to represent the canonical transformation of the Toda lattice with the restriction. Therefore, we use N refraction vectors for the parametrization of the Householder transformation. The symplectic linear transformation mixes the position and momentum. This way, by combining the two canonical transformations and stacking the block, we realize the complex transformation.

The discrete Hartley transformation plays a role of the elimination of the center of mass motion. The discrete Hartley transformation is a kind of Fourier transformation that transforms a real-valued function into a real-valued one. The discrete Hartley transformation is defined as

$$\mathrm{DHT}(q)_k \coloneqq \sum_{n=1}^{N} \sqrt{\frac{1}{N}} q_n \left(\cos\left(\frac{2\pi nk}{N}\right) + \sin\left(\frac{2\pi nk}{N}\right) \right). \tag{4.8}$$

It is equivalent to a transformation that decouples the one-dimensional harmonic oscillator chain, see Appendix. B.1.2. The discrete Hartley transformation extracts the position and momentum of the center of mass as $\mathrm{DHT}(q)_N$. We first perform the transformation in the real space and extract the coordinate of the center of mass. The remaining variables are transformed by the canonical transformation blocks. The schematic picture of the procedure is shown in the right figure in Fig. 4.2. Hereafter, the momentum and position of the center of mass are uniformly zero. It means that the action variable I_0 is always zero. The definition of I_0 is introduced in Eq. (2.31).

The parameters of the neural network are the number of canonical transformation blocks, the number of the coupling layers of RealNVP in the point transformation, the dimension of hidden layers, and the number of hidden layers in the dense neural network of realNVP. Hereafter, we fix some parameters for convenience and simplicity. The number of hidden layers in the dense network in realNVP is 5, and the dimension of the hidden layer is 30. In other words, the number of activation layers is 6. The number of coupling actions by realNVP is 4. We change the number of the canonical transformation block appropriately for each task.

4.2.3 Four types of error functions

We have the three loss functions for our procedure, L_E, L_I, L_p. In these loss functions, the function ℓ evaluates the actual loss of each part. It is not necessary to use the same error function for all the loss functions. Thus, we try to use different types of error functions in some tasks and evaluate the suitability of the functions for each task. It is not trivial that the combination of appropriate loss functions at each task is also suitable for the combined task. At least, however, if a part of the task is completely achieved, the other part is regarded as an independent task. Thus, it might be better to make the total loss function by combining suitable loss functions.

First of all, we introduce four error functions for ℓ in the loss functions. Fig. 4.3 shows the behavior of the four error functions.

First one is the standard type: the mean squared error (MSE) function introduced in Sect. 3.1. It is the most standard and famous error function. Although the function is widely used, it tends to be extremely

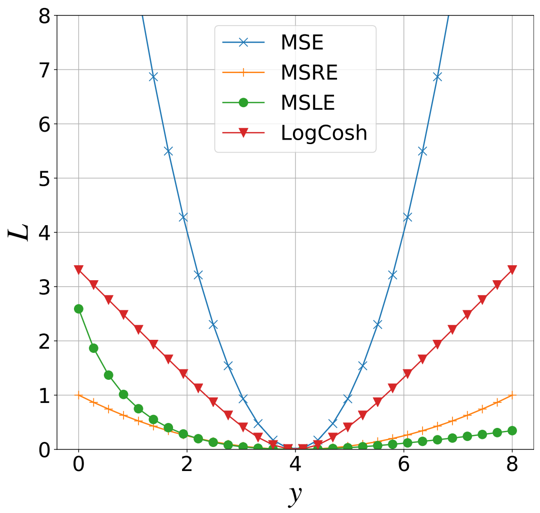


Figure 4.3: Shape of four error functions. y is the prediction. In this figure, the true value is chosen to be $y^* = 4$. MSLE increases more drastically for $y < y^*$ than $y > y^* y$. It implies that MSLE gives the neural networks more penalty in the lower predictions than higher predictions. The log-cosh function linearly increases far from the target value.

large when the difference between the prediction and target is large at initial states. Furthermore, it evaluates the whole range with the same importance. If one focuses on the relative error, it is not always suitable.

The second one is a function evaluating the relative error, called the mean squared relative error (MSRE). It is defined by

$$\ell_{\mathrm{MSRE}} = \left\langle \left| 1 - \frac{y}{y^*} \right|^2 \right\rangle_D , \tag{4.9}$$

where y is a prediction, y^* is a target value which is prepared for learning as training data, $\langle \cdot \rangle_D$ is an average of the input data. It evaluates the relative error from the true values. According to the definition, the loss function diverges when the true value is zero. Therefore, practically, one adds an extremely small number to the true value. Hereafter, we add 10^{-8} to the denominator as the small value. There are several applications using this error function [134–136]

The third one is the mean squared logarithmic error (MSLE) function. It is defined by

$$\ell_{\mathrm{MSLE}} = \left\langle |\log(y^* + 1) - \log(y + 1)|^2 \right\rangle_D . \tag{4.10}$$

Obviously, MSLE is applicable for the positive variables. The addition of one to the variables is for avoiding the divergence at zero points. This loss function is usually used for time series analysis [137–139]. As shown in Fig. 4.3, MSLE is likely to give more penalty to lower values. Thus, it makes the predictions avoid being lower than the target values.

The fourth function is the log-cosh function [140].

$$\ell_{\mathrm{log-cosh}} = \langle \log(\cosh(y^* - y)) \rangle_D . \tag{4.11}$$

The log-cosh function asymptotically behaves as MSE near the zero, while it looks like the mean absolute error function when the difference is large. In other words, when the difference $\Delta y := y^* - y$ is small, the function behaves as $(\Delta y)^2/2$. In the opposite situation, the function behaves as $|\Delta y| - \log 2$. Thus, the log-cosh function avoids divergence when the difference becomes large.

We evaluate the four types of error functions for two tasks.

One of them is learning the canonical transformation to the action-angle variables. In [79], it is revealed that the canonical transformation can be learned from the time development by unsupervised learning. The input data is exactly generated by the given system. Although they use MSE for the task in the article, it is not discussed whether MSE is most suitable or not. Through trying the various error functions by the task, we evaluate the appropriate type of error function for learning canonical transformation. The task is represented by the terms introduced in the diagram Fig. 4.2:

$$L = \frac{1}{NN_{\text{time}}} \sum_{i=1}^{N} \sum_{k=1}^{N_{\text{time}}} \left\langle \ell \left(I_i(f(x)) - I_i(f \circ \phi^{t_k}_{\text{exact}}(x)) \right) \right\rangle_D , \tag{4.12}$$

where $\phi^t_{\text{exact}}(x)$ is the time development from x by the exact Hamiltonian and f is the neural canonical transformation. More precisely, the action variables are obtained by the composition of canonical transformation $\mathscr{F} \circ f$, but we omit it for simplicity. We noted that we try the three error functions, MSE, MSLE, and log-cosh functions, for the task. The reason is that the input is the difference between predicted action variables at two points. It does not compare the prediction with true data but with another prediction. Thus, the relative error does not make sense in the present case.

The other task is learning the potential function by supervised learning. It is a relatively simple task. We input points in the real space into the neural networks and compare predictions and thier energy calculated from the exact Hamiltonian. We evaluate all of the types of error functions in this task. The task is described in terms of the diagram as the following:

$$L = \langle \ell \left(H(x) - E_{\text{exact}} \right) \rangle_D , \tag{4.13}$$

where E_{E} is the exact value of the energy, and H is the Hamiltonian constructed by the neural networks.

In the two tasks, we sample the training data from the Boltzmann distribution characterized by the Hamiltonian. Since we consider a situation that we know the true Hamiltonian in the two tasks, we can obtain the data from the thermodynamic distribution. Through the condition, we introduce the temperature into the training data. Although it is not necessary to introduce the thermodynamic properties, it provides us the parameter which controls the spread of the data.

To evaluate the loss function, we introduce two scores for the two tasks. They are a score about the prediction of energy and a score about conserving the action variables. As mentioned below, they are different from the loss functions introduced above. We introduce the different functions from the above because we evaluate the loss functions by the same criterion. Obviously, when the neural networks achieve the minimum value of the loss function, there is no difference in the loss functions. There are, however, situations where the different loss function reaches a different point, which is not the true minimum. In such a situation, we have to evaluate the trained neural networks by the same criterion. Thus, we introduce the scores which are different from the loss functions.

For the energy prediction, the score η_E is defined by

$$\eta_E \coloneqq \left\langle \left| 1 - \frac{E}{E_{\text{exact}}} \right| \right\rangle_{D_{\text{Valid}}} , \tag{4.14}$$

where $\langle \cdot \rangle_{D_{\text{Valid}}}$ means an average of the validation data. The validation data is sampled independently from the training data. The evaluation by the validation data indicates the generalization performance of the trained neural network. When the score η_E is smaller, the prediction is more accurate.

We can evaluate the conservation of the action variables for the trained neural potential function and canonical transformation. For these two cases, we introduce the score of the conservation of the action variables, denoted as η_C, defined as

$$\eta_C \coloneqq \frac{1}{N-1} \sum_{i=1}^{N-1} \left\langle \frac{\sqrt{\left\langle I_i^2(\phi^t(x)) \right\rangle_{\text{time}} - \langle I_i(\phi^t(x)) \rangle^2_{\text{time}}}}{\langle I_i(\phi^t(x)) \rangle_{\text{time}}} \right\rangle_{D_{\text{Valid}}} , \tag{4.15}$$

where $\langle \cdot \rangle_{\text{time}}$ is the time average defined by

$$\langle g \rangle_{\text{time}} \coloneqq \frac{1}{N_{\text{time}}} \sum_{n=1}^{N_{\text{time}}} g(t_k), \tag{4.16}$$

where N_{time} is the number of time points. The numerator is the variance of the action variables for the time. The denominator is the time-averaged mean value of the action variables. Therefore, the score means the relative variation of the action variables under time development. The definition of the score η_C includes three cases. For the trained neural potential function, ϕ^t is operated by the trained neural networks. On the other hand, the action variables are calculated by the exact form. For instance, the exact action variables of the Toda lattice is obtained by Eq. (2.43). For a harmonic chain, the actions are calculated by Eq. (2.31). Therefore, we evaluate how the trained neural potential function conserves the action variables. For the trained neural canonical transformation, the time development is performed by the exact Hamiltonian. Thus, we evaluate the trained neural canonical transformation by the conservation of the action variables. As well as the score η_E, the conservation score η_C is smaller, the conservation is better. The third case is that all the components are generated by the trained neural networks. It means that the trajectories transformed by the canonical transformation are generated from the trained neural potential function. Moreover, the canonical transformation to obtain the action variables is also a trained neural network. In other words, η_C is calculated without any exact solutions. Thus, we can calculate η_C for the system, which is unknown but obtained by neural networks. Hereafter, we denote η_C with such specific condition as η_{NN}

We note that the prediction error of the action variables is not always defined. The reason is that the action variables are not determined uniquely, as mentioned in Sect. 2.2.1. Thus, we evaluate the canonical transformation as the conservation of the predicted action variables.

4.2.4 Evaluation of the error functions for the canonical transformation

We evaluate the four error functions introduced above by the Toda lattice. The Toda lattice is one of the most famous integrable systems and rich properties. Though the harmonic oscillator chain is also the most famous system, it is too simple to test the neural networks.

First, we evaluate the three types of error functions, MSE, MSLE, and log-cosh, for the learning canonical transformation task. In the task, we use the loss function defined by Eq. (4.12). We sample the training and validation data from the Boltzmann distribution by the Hamiltonian Monte Carlo (HMC) [141, 142]. The detail of the algorithm and validation of the parameters in the method is presented in Appendix. C. Here, we sample the data from the distribution with a temperature $T = 2.0$. The number of training data is 4000, and the number of validation data is the same. We use the mini-batch gradient descent and the Adam algorithm as the optimizer, which is introduced in Sect. 3.1. The size of the mini-batch is 200. Thus, the number of mini-batches is 20 for the training data and the validation data. We use the loss value per mini-batch to evaluate the learning progress for the convenience of the implementation and simplicity. We define the loss value per mini-batch and the standard deviation as

$$\begin{aligned} \langle L_{\mathrm{MB}} \rangle &:= \frac{1}{N_{\mathrm{MB}}} \sum_{k=1} L_{\mathrm{MB},k}, \\ \sigma_{\mathrm{MB}} &:= \sqrt{\frac{1}{N_{\mathrm{MB}}} \sum_{k=1} \left(L_{\mathrm{MB},k}\right)^2 - \frac{N_{\mathrm{MB}}}{N_{\mathrm{MB}}-1} \langle L_{\mathrm{MB}} \rangle^2}, \end{aligned} \tag{4.17}$$

where N_{MB} is the number of the mini-batches, and $L_{\mathrm{MB},k}$ is the loss value in the k-th mini-batch. We note that the standard deviations have a non-negligible statistical error due to the small N_{MB}. Thus, the quantity roughly estimates the deviation of the loss values. The number of epochs is chosen to be 10^3. Here, we call the period where all the training data is used as a mini-batch one epoch. In other words, the neural networks are updated for 20 times during one epoch. For this task, we use the learning rate scheduler for efficiency. The learning rate scheduler reduces the learning rate if the loss function is small enough and does not show any improvement. We reduce the learning rate to 80% when the loss value does not decrease more than 1% of the lowest value in 10 epochs. The initial learning rate is chosen to be 10^{-3}. Hereafter, unless otherwise noted, the initial learning rate is always 10^{-3}. Furthermore, the condition for the time development in the loss function is as follows: the time interval dt is 0.05, and the total time t is 5.0. It means the number of time steps is 100. Hereafter, we set the two parameters of

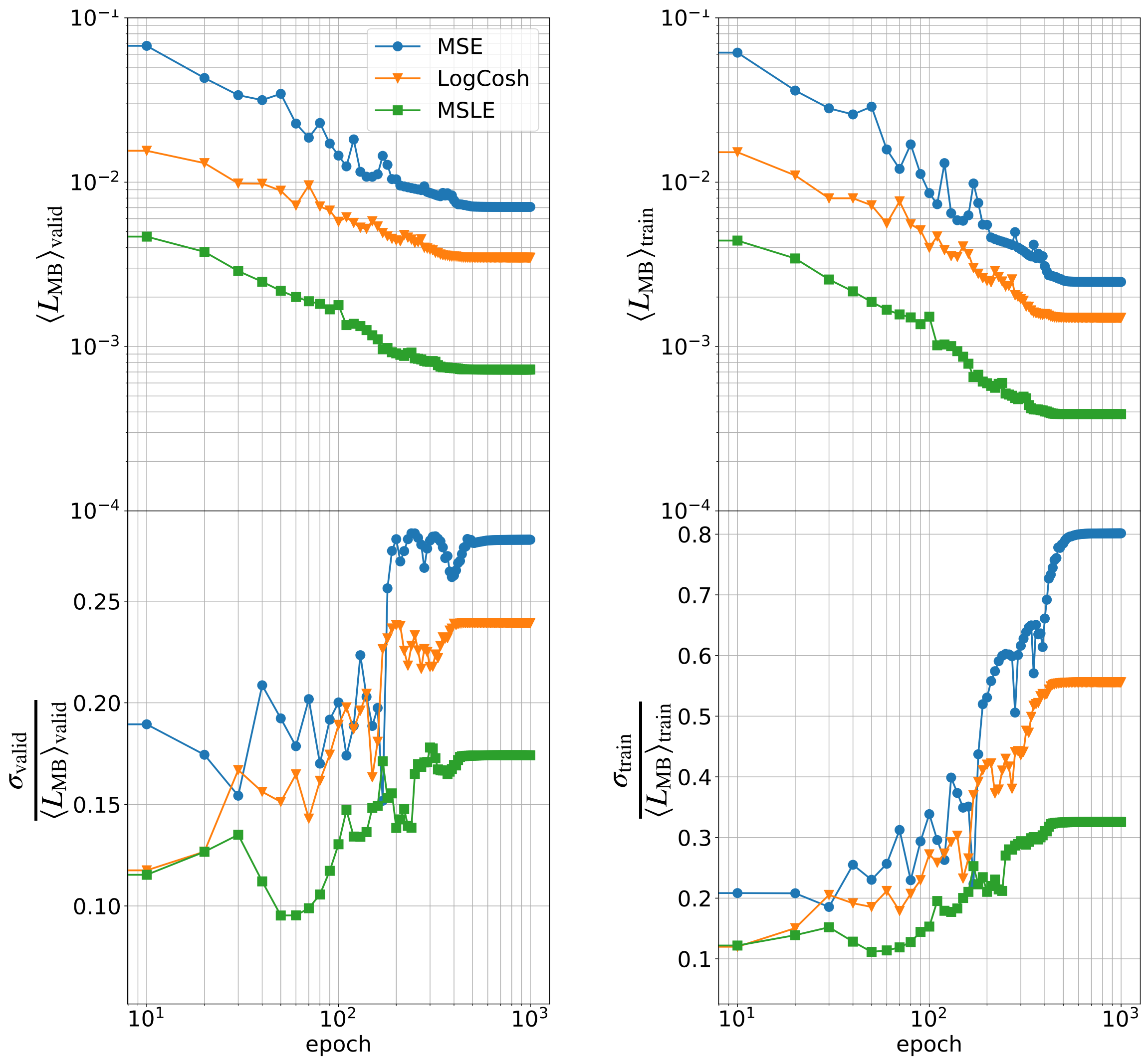


Figure 4.4: Change of loss values in the process for learning the canonical transformation at different types of error function. We use MSE (blue circle), log-cosh error (orange down triangle), and MSLE (green square). The vertical axes are the loss value per mini-batch of validation and training in the upper figures, respectively. We can see the convergence at the final learning stage. For simplicity, we omit the error bar from the plot. The vertical axes are the ratio of the standard deviation and the loss value in the lower figures. The lower left figure shows the validation loss ratio, and the lower right is the train one. They show the variations of the loss values in the mini-batches.

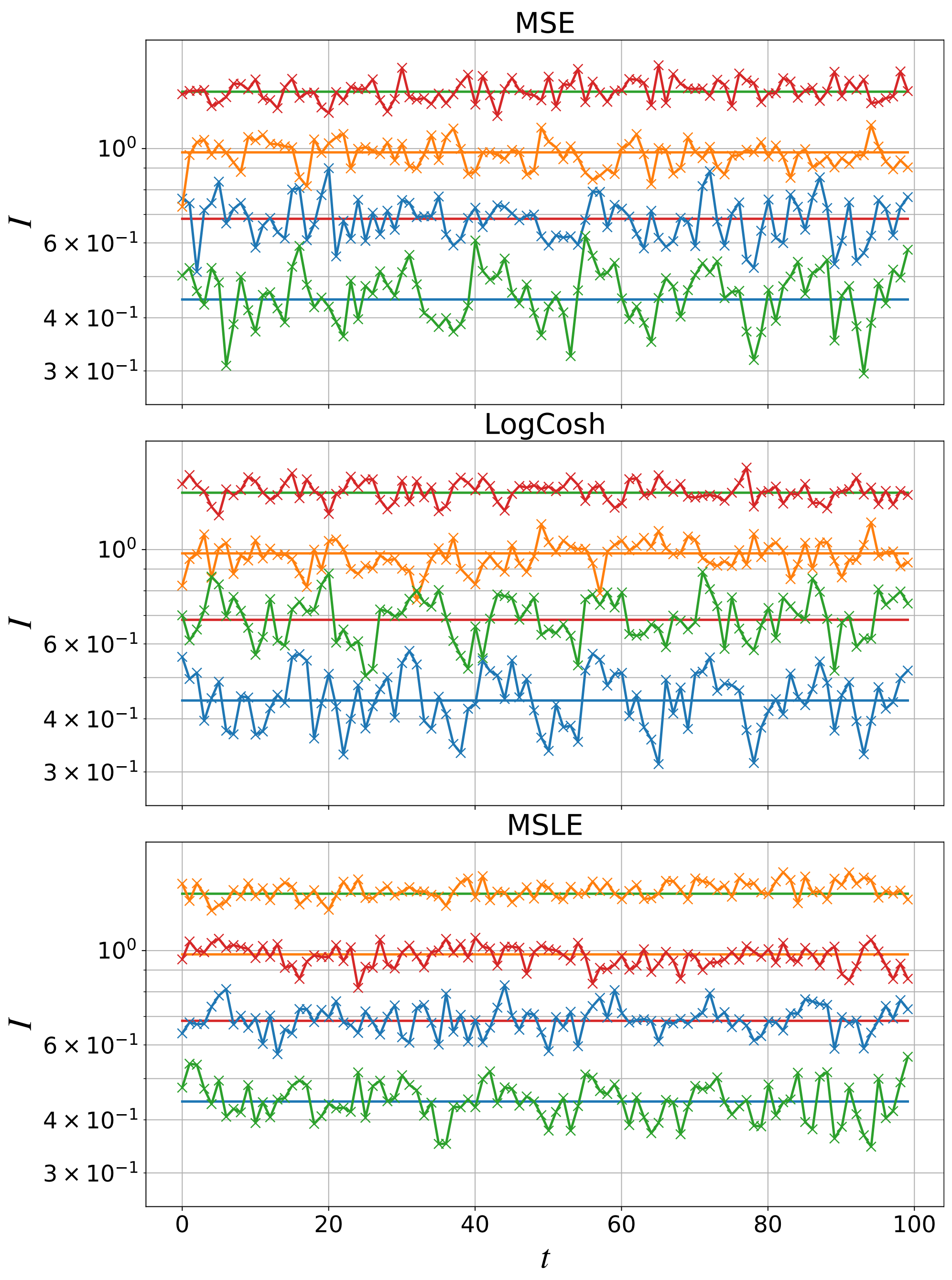


Figure 4.5: Time series of action variables predicted by the trained neural canonical translation. The colors correspond to the index of action variables. The relations are as follows: I_1 (blue), I_2(orange), I_3(green), I_4(red). The neural networks predict the line plotted with crosses. The straight line shows the values calculated by Eq. (2.43). The color correspondence is the same as the predictions by the neural network. We omit the plot of I_0, which corresponds to the momentum of the center of motion because it is not a prediction. The prediction by the neural network trained by MSLE seems to be more accurate than the others. The initial state is sampled from the Boltzmann distinguish with $T = 2$. The energy of the initial state is $E = 5.99$. The scores are 0.140 (MSE), 0.145 (log-cosh), 0.124 (MSLE).

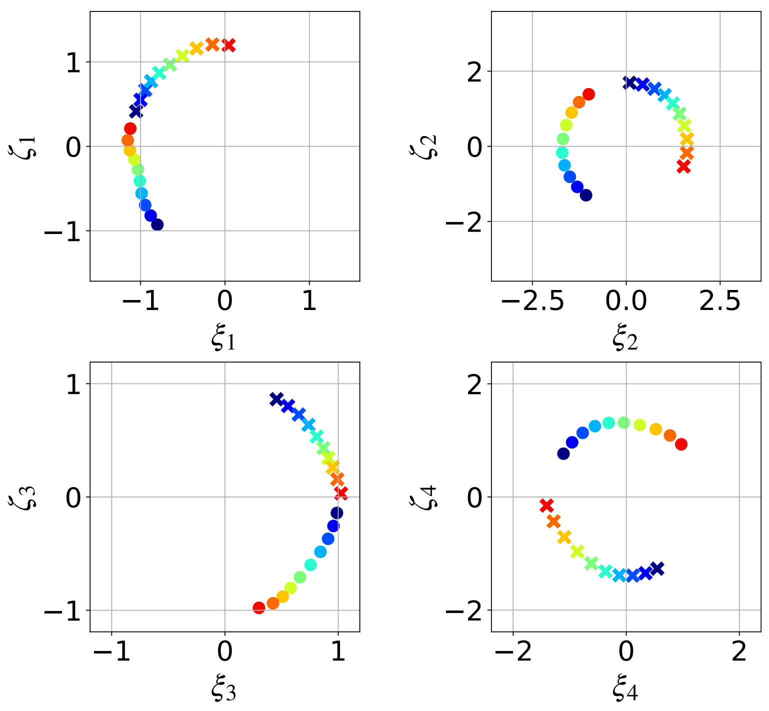


Figure 4.6: Time series in latent space predicted by the MSLE trained neural network. Colored circles show the transformed data by the neural network. The radius of the time series corresponds to the action variables. Conversely, the colored cross shows the points transformed from the cyclic permuted data. The cyclic permutation causes the phase shift of the angle. The jet colormap stands for the time points. Cold colors correspond to the initial time points and warm colors the final states in the period $t = [0.0, 1.0]$.

the Toda lattice as $J = 1$ and $\alpha = 1$, and the number of particles is chosen to be 5. The number of time points for comparing the difference of action variables in the time development is 5. In other words, we use the action-angle variables at $t = 1.0, 2.0, \ldots, 5.0$ in the loss function.

The results are as follows. The change of the loss value is shown in Fig. 4.4. As the upper figure shows, all the processes exhibit convergence. It means the neural networks achieve the possibly best state of the loss function and training data. The upper left figure shows the validation loss and the upper right shows the training one. For all the cases, the validation loss is larger than the training one. In general, such a phenomenon is observed. The neural networks know just the training data, but the validation data is unknown for them. Thus, it is natural that the model fits the training data better than the validation one. We need more care when the validation loss becomes worse, even if the training loss is improved. In such a situation, the neural network fits the training data excessively, and its generalization is failed. In the results shown in Fig. 4.4, the validation loss converges as well as the training loss. Therefore, in the present simulation, the learning processes are expected to be done without overfitting.

The lower figures in Fig. 4.4 shows the ratio of the derivation of loss values. As mentioned above, the standard deviations have a non-negligible error. In this sense, the values roughly imply the variations. For the MSLE case, the standard deviation is smaller than the other cases. It might imply that the generalization performance of MSLE is better than the others. However, to conclude that MSLE is better for the generalization, one must compare various situations by changing the mini-batch size and the number of mini-batches. Here, our aim for observing the loss change is to confirm the learning progress. In this sense, it is enough to observe the convergence of the loss values. Furthermore, we evaluate the performance of the trained neural networks by scores defined by Eq. (4.15) and Eq. (4.14). At least, if the validation loss converges to a smaller value, we can expect that the performance of the trained neural network is high enough. Thus, we have no more discussion of the behavior of the loss change.

Next, we check predictions by the trained neural networks. Fig. 4.5 shows the prediction of the action variables for the state sampled from the Boltzmann distribution. The input data is a time series generated by the time development of the exact Hamiltonian. The time interval dt is 0.01. The initial state is sampled from the Boltzmann distribution at $T = 2$. It is surprising that the neural networks well predict the action variables of longer time series than the training data. The neural networks learn the canonical transformation by the time series up to $t = 5$. In the figure, the prediction by the neural network trained with MSLE seems to be more accurate. We note that the correspondence between the index of the analytical calculation and the prediction is not necessary to hold. The learning process is unsupervised and requires just the conservation of the action variables. If one gives the analytically calculated quantities for the learning procedure, the trained canonical transformation provides the prediction whose index is equal to the analytical one.

Here, we observe properties of the canonical transformation. Fig. 4.6 shows the time series of the latent space produced by the trained neural network. The points show the rotation, and the radii correspond to the action variables. We also plot the time series whose initial state is cyclic permuted. It means that the initial state is rotated as $q_1 \to q_2, q_2 \to q_3, \ldots, q_5 \to q_1$. For the harmonic oscillators, the transformation causes the phase shift to the normal modes. By the trained canonical transformation, the action-angle variables of the Toda lattice also exhibit the same properties. In this sense, the canonical transformation of the Toda lattice is regarded as the normal modes, but nonlinear.

Next, we evaluate the performance of the loss functions by the score η_C. The score is defined by Eq. (4.15). For the evaluation of the score, we use the following parameters: the development time is $t = 100$, and the time interval is $dt = 0.01$. Hence, the number of time steps is 10^4. The number of the sampled time points from the time development is 1000. It means the time points are sampled from the time series in each $t = 0.1$. The number of initial states is 10^4. They are sampled from the Boltzmann distribution by HMC. The parameters are the same as the training data and validation data, but the random seed is different. The results are shown in Table. 4.2.4. The conservation score η_C means the relative deviations of the action variables. The performance is better if the score is smaller. Consequently, the MSLE error function is expected to be suitable for the task. Hereafter, we apply MSLE for learning action-angle variables. We mention that the performance possibly becomes better if one stacks more

	MSE	LogCosh	MSLE
η_C	0.140	0.145	0.124

Table 4.1: Conservation score of the three error functions.

layers in the neural network. If the number of the block increases up to 14, the performance becomes better. However, as mentioned below, for the combined task, which is our aim, the large number of the blocks causes divergence of the loss value at the initial training stage respecting the target systems.

4.2.5 Evaluation of the error functions for the potential funciton

Next, we evaluate the error functions for learning the potential functions. We use the four functions, MSE, MSLE, MSRE, and log-cosh error function. For this task, the loss function is defined by Eq. (4.13). The condition of the learning process is as follows: the number of training data is 10^4 as well as the validation data. The size of the mini-batch is 2000. It means the update of the neural network is performed 5 times per epoch. We train the neural networks for 300. We evaluate the four types of the error function by two neural network structures, residual and dense neural network. Furthermore, we fix the interaction form of the potential function on the nearest-neighbor interaction. We change the number of hidden layers denoted by N_{hl}.

The results are shown in Fig. 4.7 and Fig. 4.8. For all the best cases, the difference between the validation loss and training loss converges to zero. It means the trained potential function predicts the true value well even if the input data is not included in the training data. Thus, the generalization performance is expected to be high enough. In the case of the dense neural network, when the number of hidden layers increases, the loss value decreases more. Therefore, for the task, it is not necessary to tune the parameters of the neural networks. Conversely, for the residual neural network, deeper neural networks do not always exhibit better performance. To make matters worse, they have the worst performance.

Before we observe the behavior of the trained potential function, we evaluate the trained potential function by the conservation score η_C and energy prediction score η_E. They are defined by Eq. (4.15) and Eq. (4.14), respectively. We use different parameters from the case of the canonical transformation learning due to the computational cost. The time of development is $t = 100$, and the time interval is $dt = 0.01$. Hence, the number of time steps is 10^4. The number of sampled time points from the time development is 100. It means the time points are sampled from the time series in each $t = 1.0$. The number of initial states is 10^3. The results are shown in Table. 4.2.5. For both cases, the log-cosh error function performs the best.

Dense	MSE	LogCosh	MSLE	MSRE
η_E	6.0×10^{-4}	3.0×10^{-4}	4.4×10^{-4}	5.6×10^{-4}
η_C	0.035	0.014	0.024	0.024
Residual				
η_E	7.9×10^{-4}	5.1×10^{-4}	6.3×10^{-4}	8.2×10^{-4}
η_C	0.042	0.020	0.036	0.042

Table 4.2: Conservation and prediction scores of the trained potential function with four types of error functions. In the upper table, the potential functions are represented by dense neural networks. In the lower table, parametrization is performed by residual neural networks.

We observe the behavior of the trained potential function. We use the dense neural network and log-cosh error function. Furthermore, we train the neural networks by using the data sampled at a different temperature. The trained potential functions are shown in the left figure of Fig. 4.9. We can observe the linearly growing difference between the true Toda potential and the neural networks. The reason is that the lattice has periodic boundary conditions. The conditions make the linear potential vanish in the total potential energy. Thus, there is such uncertainty of linear functions. We can easily remove the freedom

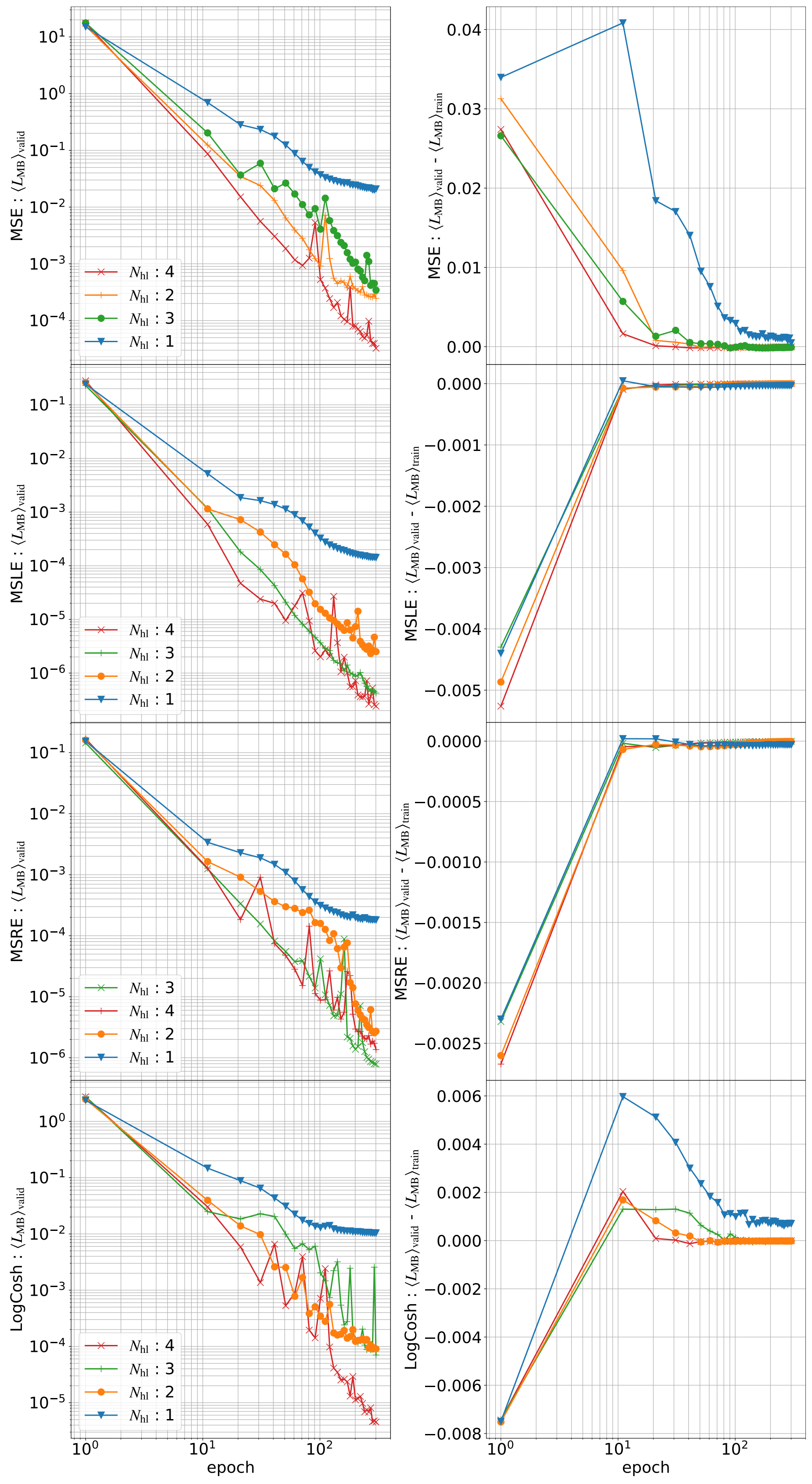


Figure 4.7: Change of loss values in the process for learning the potential function with different type error functions. We use the dense neural networks for the parametrization. The right figures show the loss change of the training data. The left figures show the loss change of the validation data. If the number of hidden layers increases in the dense neural network case, the loss values decrease more.

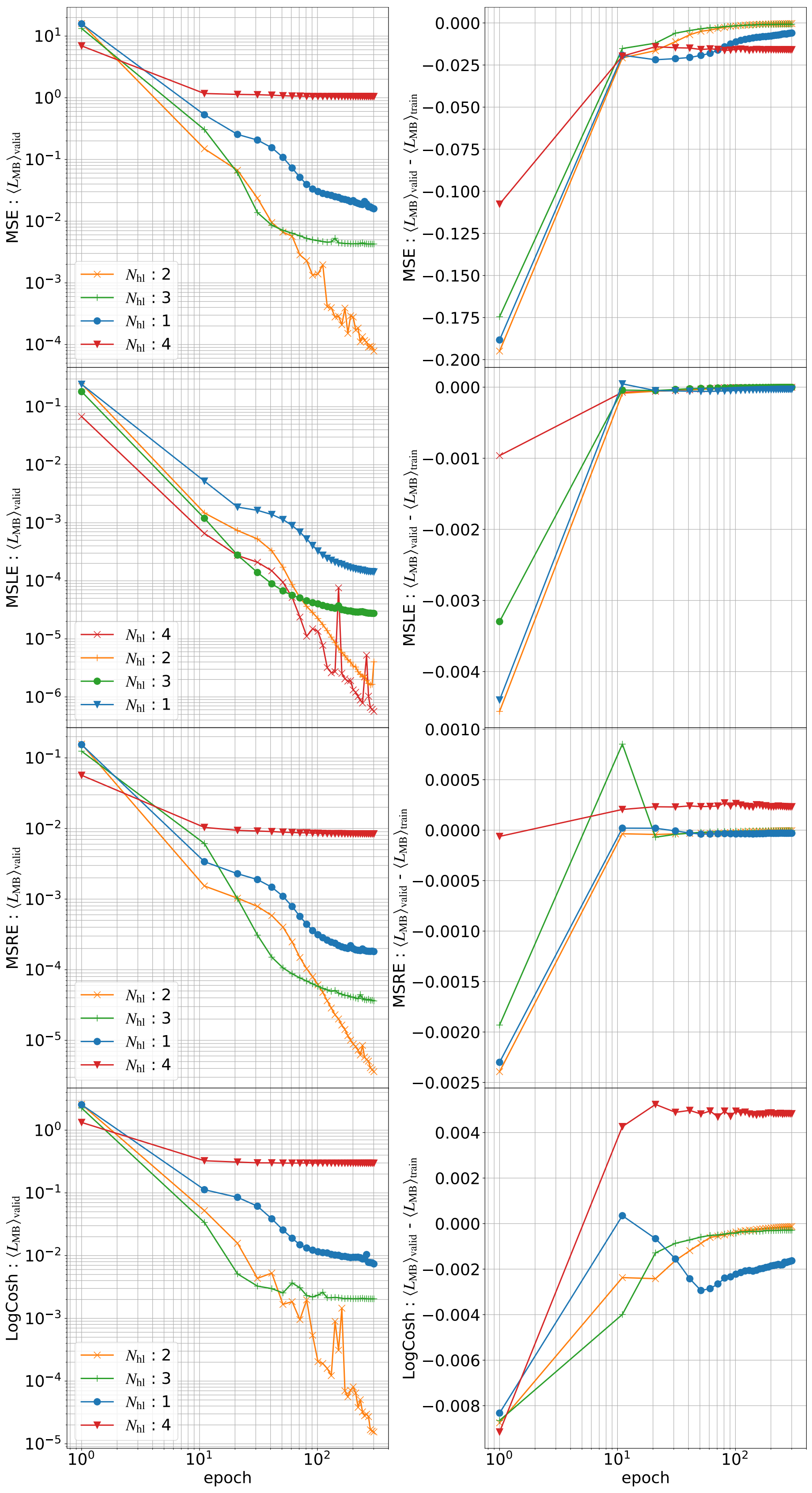


Figure 4.8: Change of loss values in the process for learning the potential function with different type error functions. We use the residual neural networks for the parametrization. The right figures show the loss change of the training data. The left figures show the loss change of the validation data.

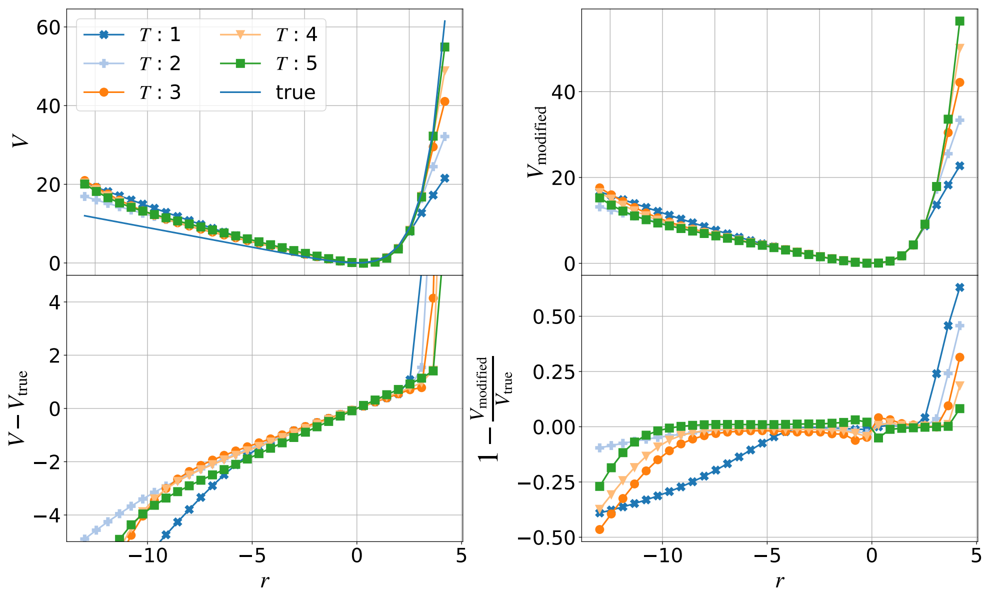


Figure 4.9: Trained potential function for different training data with respect to the temperature. The true functions is the Toda lattice potential, $e^x - x - 1$. In the upper left figure, the trained potential functions are shown. The lower left figure shows the difference from the true function. The linear dependence of the difference is observed. In the upper-right figure, the modified trained potential functions with different temperatures are shown. The neural networks remove the linear function by the derivative at zero. The lower right figure shows the difference between the modified potential function and the true potential function. The neural networks trained at higher temperatures predict the true potential in a broader area.

by using the derivative of the potential at zero. Thus, the modified traind potential function is given by

$$V_{\text{modified}}(r) = V(r) - \left.\frac{\partial V(r)}{\partial r}\right|_{r=0} r. \tag{4.18}$$

The modified trained potential functions are the shown in right figure of Fig. 4.9 The neural networks trained at the higher temperatures predict the true potential well in a wider area. The behavior is natural because the high-temperature distribution provides data distributed more broadly. Thus, if the neural network learned the data successfully, the prediction is expected to be applicable in the broader area.

Based on the results of the above tests, we use MSLE for the canonical transformation and the log-cosh function for the potential function in the following. As mentioned in the first part of Sect. 4.2.3, however, it is not trivial that the combined loss function also works for the combined task. Thus, we always consider modifying the loss function for each task.

4.2.6 Tests and applications: simple loss function defined by Eq. (4.2)

First of all, we try the simple loss function defined by Eq. (4.2). According to the above evaluations, we use MSLE as ℓ in the loss function. In this approach, we use the data set defined in the real space. There are some methods for generating the data set. Here, we use a Gaussian distribution for the training data set. Concretely, we set the mean value and standard deviation to 0 and 2, respectively. The size of the training data and the validation data are 10^4. The size of the mini-batch is 2000, and the initial learning rate is 10^{-3}. The length of time development is $t = 100$, and the number of time points for the loss calculation is 5. By the results of this approach, we see the necessity of the additional condition for finding the non-trivial integrable systems.

The loss function is expected to extract the integrable systems by unsupervised learning. We note that the simple loss function needs a small modification. In MSLE, the large norm of the coordinates in

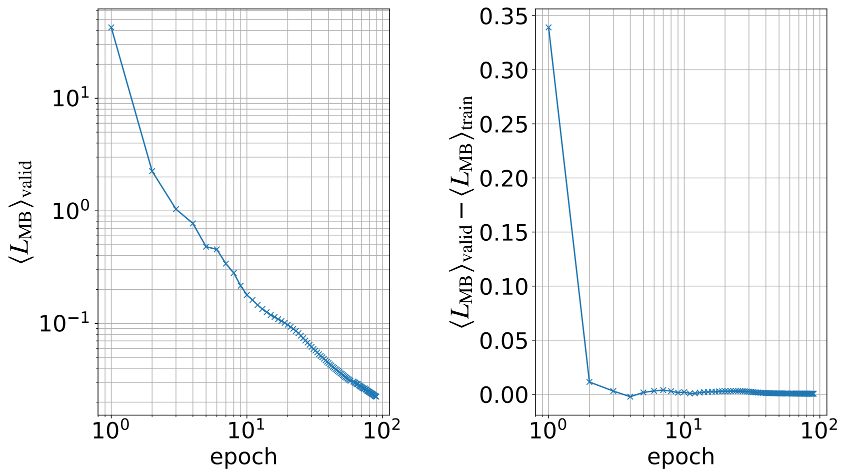


Figure 4.10: Change of the loss in the first example. We train the neural network for 90 epochs. The validation loss does not converge. The difference between validation and training loss is small enough.

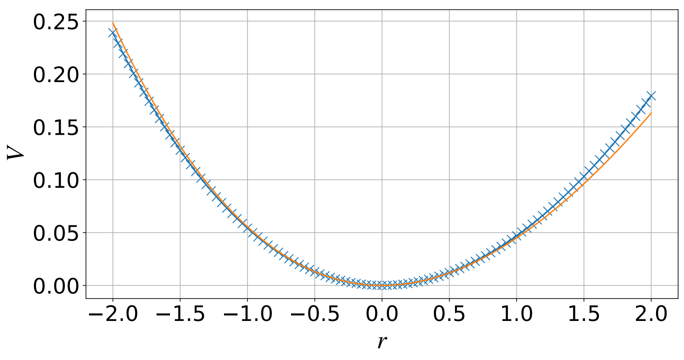


Figure 4.11: Obtained potential function in the first example. The crosses show the trained potential function after the removal of the constant and the linear function. The orange line represents the fitting function $e^{-0.31r} - 1 + 0.31r$.

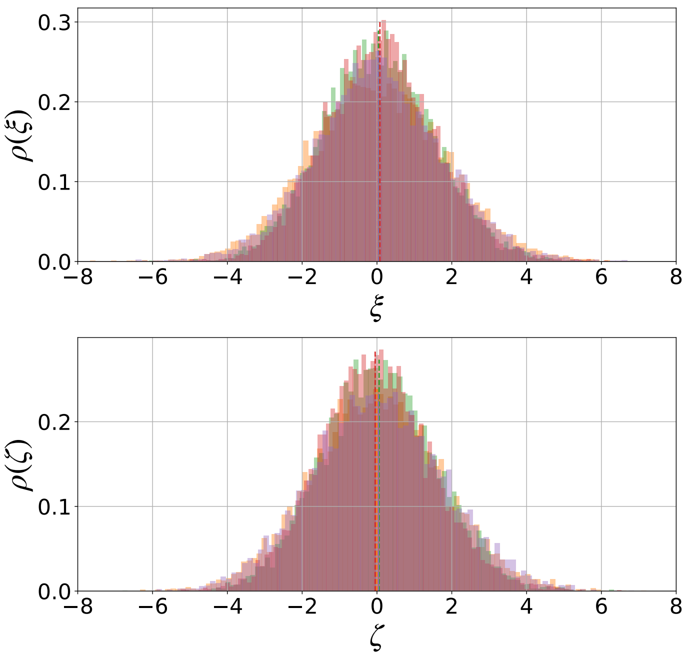


Figure 4.12: A histogram of the latent space transformed from the validation data set by the trained canonical transformation. Four histograms are plotted but overlapping as the mean values are almost the same. The mean values are $0.027(i = 1), 0.048(i = 2), 0.076(i = 3), 0.022(i = 4)$.

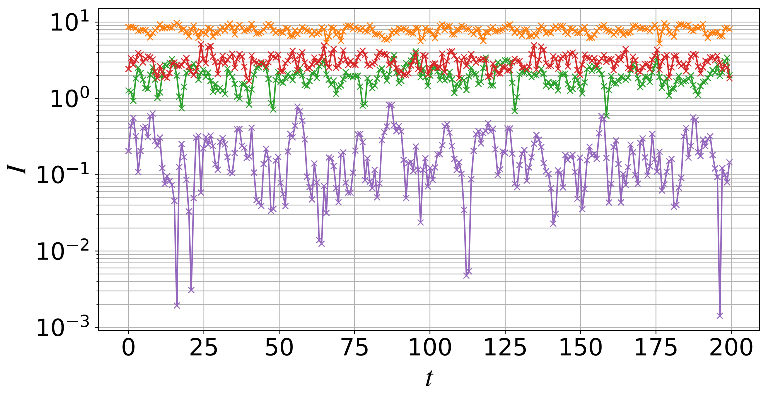


Figure 4.13: Time series of the action variables obtained by the trained canonical transformation. The trajectories in the real space are generated by the trained potential function. The relations of the colors are as follows: I_1 (blue), I_2 (orange), I_3 (green), I_4 (red). The score $\eta_{\rm NN}$ is 0.240.

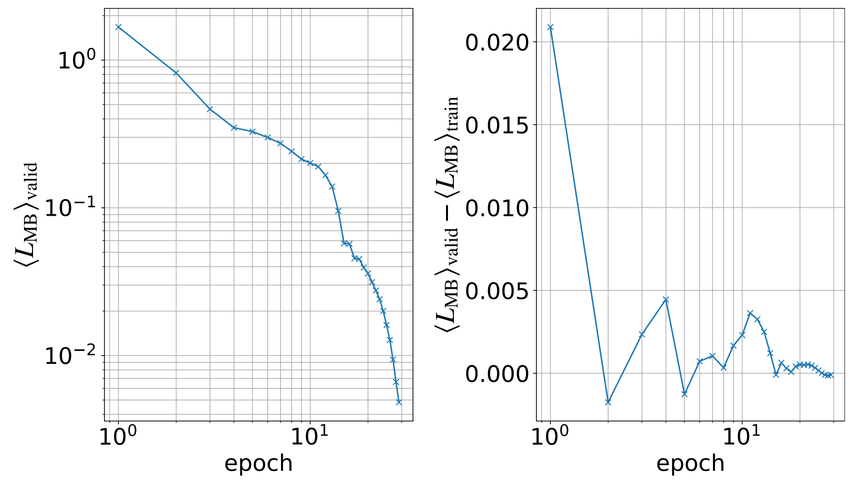


Figure 4.14: Change of the loss in the second example. We train the neural network for 29 epochs. The validation loss does not converge. The difference between validation and training loss is small enough.

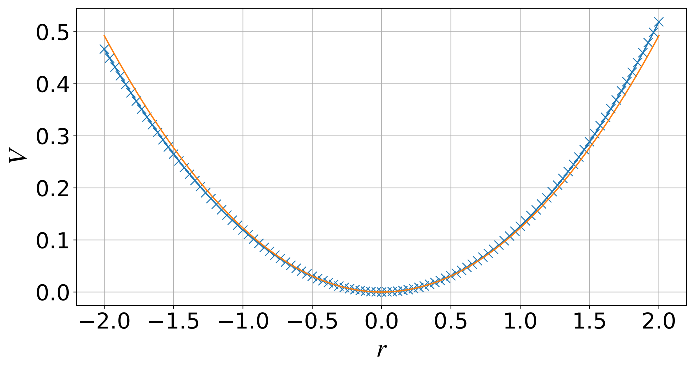


Figure 4.15: Obtained potential function in the second example. The crosses show the trained potential function after the removal of the constant and the linear function. The orange line represents the fitting function $0.12r^2$.

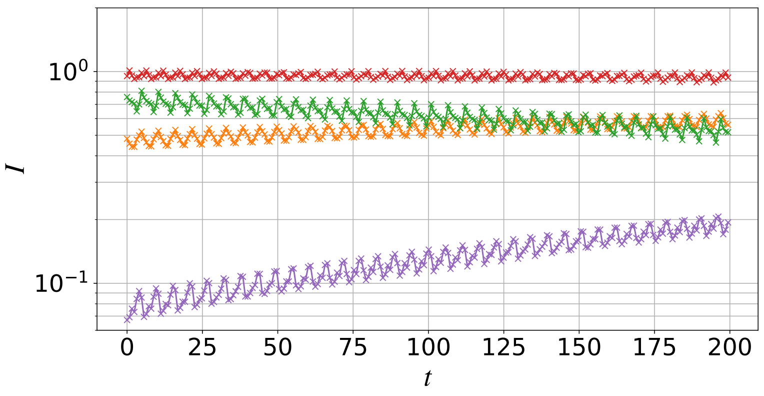


Figure 4.16: Time series of the action variables obtained by the trained canonical transformation. The trajectories in the real space are generated by the trained potential function. The relations of the colors are as follows: I_1 (blue), I_2 (orange), I_3 (green), I_4 (red). The score η_{NN} is 0.072.

the latent space has more advantage because MSLE evaluates the relative error of the action variables. Furthermore, the constant potential function also has advantages for the loss function. The reason is that the motion driven by the constant potential function can be regarded as an infinitely long periodical motion. Thus, we modify the loss function to suppress the behavior as follows:

$$L = L_I + \lambda L_{(\xi,\zeta)}, \tag{4.19}$$

$$L_{(\xi,\zeta)} \coloneqq \sum_{i=1}^{N} \left(\ell(\langle \xi_i \rangle_D) + \ell(\langle \zeta_i \rangle_D)\right), \tag{4.20}$$

where ξ_i and ζ_i are the coordinates of the latent space, and λ is a parameter for tuning the strength of the loss function. The loss function requires centering the rotation of the latent space by the penalty for the mean value of ξ_i and ζ_i. We mention that we assume that the data set has enough periodic trajectories in the latent space. If we remove the loss function for the latent space, it is difficult to determine whether we obtain the canonical transformation extracting the valid action variables or mapping to too large norm coordinates. The total loss function is thus given by

$$\begin{aligned} L &= L_I + \lambda L_{(\xi,\zeta)} \\ L_I &= \frac{1}{N N_{\text{time}}} \sum_{i=1}^{N} \sum_{k=1}^{N_{\text{time}}} \left\langle |\log(I_i(t_0)+1) - \log(I_i(t_k)+1)|^2 \right\rangle_D, \\ L_{(\xi,\zeta)} &\coloneqq \sum_{i=1}^{N} \left(\ell(\langle \xi_i \rangle_D) + \ell(\langle \zeta_i \rangle_D)\right). \end{aligned} \tag{4.21}$$

We try to train neural networks by the loss function under various conditions. We change the activation function, the structure of the potential function, the lattice structure, and the random seed. Here, we present some suggestive results. We note that the neural networks assuming the multi-body interaction do not exhibit good performance. That is, the loss function does not decrease.

First, we introduce the result obtained by the neural networks parametrizing $V_{\text{chain}}(q)$. We set the constant $\lambda = 10^{-3}$. The change of the loss is shown in Fig. 4.10. Although the validation loss does not converge, the loss value becomes small. Fig. 4.11 shows the obtained potential function. We note that the plot shows the modified function, which is the trained potential after removing a linear function and a constant. By assuming an exponential function $ae^{-br} - a + abr$ as the fitting function, we obtain $a = 1, b = 0.31$. This result means that the trained potential function is close to the Toda lattice, but the amplitude of the potential function is too small. As mentioned above, the loss function tends to make the neural potential function small. Fig. 4.12 shows the distribution of the latent space obtained by the trained canonical transformation. Overlapping of the histograms shows that the loss function for the latent space works well. We observe the action variables obtained by the trained canonical transformation. Fig. 4.13 shows the behavior of the action variables. The action variables are obtained by the trained canonical transformation from the trajectories generated by the obtained potential function. The neural networks are trained by the data with a time period $t = 100$. The predicted action variables fluctuate strongly. In fact, the conservation score η_{NN} is 0.240. The conditions are as follows: $t = 100$, the number of time points is 10^3, and the number of initial states is 10^4.

The second example is the neural networks parametrizing $V_{\text{fully}}(q)$. We set the constant $\lambda = 10^{-4}$. The loss change is shown in Fig. 4.14. The validation loss changes drastically. Fig. 4.15 shows the obtained potential function. We assume ax^2 as the fitting function and obtaine $a = 0.12$. We omit the latent space coordinates histogram because the result is almost similar to the first example. We observe the action variables obtained by the trained canonical transformation. Fig. 4.16 shows the behavior of the action variables. Similar to the first example, the action variables are obtained by the trained canonical transformation from the trajectories generated by the obtained potential function. The neural networks are trained the data with a time period $t = 100$. The fluctuation of the predicted action variables is smaller than the first example. In fact, the conservation score η_{NN} is 0.072. The conditions are as follows: $t = 100$, the number of time points is 10^3, and the number of initial states is 10^4.

As we observe in the behavior of the trained neural networks by the loss function defined by Eq. (4.21), the approach is not appropriate for finding integrable systems. The essential point is that the state achieving the minimum of the loss function is difficult to be found. In the first example, the potential function extracts the shape of the Toda potential. The second example exhibits simple fully-connected harmonic oscillators. Although there is a probability that the Toda potential or the other potential is extracted by the training process, it is hard to control the learning process. Thus, it is appropriate to add more information about the desirable systems to the learning process.

4.2.7 Tests and applications: complex loss function defined by Eq. (4.6)

Nest, we consider an approach using the loss function defined by Eq. (4.6). We apply the approach to learn the Toda lattice. As mentioned above, $K(I)$ of the Toda lattice is not obtained analytically, and just the lower expansion is known (see Eq. (2.45)). Thus, we create the data set constructed by the pair of action variables and the energy. For the learning process using the loss function, $K(I)$ provides us the data set and the energy. Since we know the transformation to the action variables by Eq. (2.43), we prepare the action variable and corresponding energy from the original real space coordinates instead of sampling using $K(I)$. Concretely, we sample the points in the real space by the Boltzmann distribution. The Boltzmann distribution is described by the real space Hamilton $H(x)$, and the data set is created by the HMC. Then, we transform the points into the action variables and calculate the energy at the same point. The pair of energy and action variables play a role of data produced by $K(I)$. Through the method, we perform the learning process for the Toda lattice.

We use the different type of error function for each part of Eq. (4.6). According to the loss function analysis, we use MSLE for the loss function of action variables, L_I. On the other hand, the log-cosh error function is used for the loss function of energy, L_E. Here, we emphasize that it is not trivial that the combination of the best error functions is appropriate for the combined task. We reveal that the loss function using the MSLE for L_E shows better performance than one using the log-cosh error function for the task. For the loss of momentum, the log-cosh error function or MSE is applicable. The reason is that MSLE can treat only the positive quantity, and MSRE accepts the non-zero quantity. The mean value of the momentum is required to be zero. Thus, the two error functions are not appropriate. In this research, we use the log-cosh error function for the momentum loss. We expect that the log-cosh error function suppresses the divergence by its asymptotic behavior. Before showing the results, we review our loss function. The concrete representation of the loss function is given by

$$
\begin{aligned}
L &= L_I + L_E + L_p, \\
L_I &= \frac{1}{N N_{\text{time}}} \sum_{i=1}^{N} \sum_{k=1}^{N_{\text{time}}} \left\langle |\log(I_i(t_0)+1) - \log(I_i(t_k)+1)|^2 \right\rangle_D , \\
L_E &= \langle \log(\cosh(H(x) - E_{\text{exact}})) \rangle_D , \\
L_p &= \frac{1}{N} \sum_{i=1}^{N} \left\{ \log\cosh(\langle p_i \rangle_D) + \log\cosh(\left\langle p_i^2 \right\rangle_D - \left\langle p_i^2 \right\rangle_{\text{thermal}}) \right\} ,
\end{aligned}
\tag{4.22}
$$

Moreover, we try several structures of the neural network for the learning process. We use the residual and dense neural networks and two activation functions, softplus and SiLU. We construct the canonical transformation by stacking 10 canonical transformation blocks. The potential function is constructed by 4 hidden layers, and the dimension of the hidden layers is 128. To create the training and validation data, we chose the temperature to be 2 for the Boltzmann distribution. The size of the training data and validation data is 10^5. The time series for calculating the action variable loss is $t = 5$, and the number of time points is 5. The time interval is 0.05.

Below, we show three results about the learning of the Toda lattice. The first two results are that the learning process achieves finding the Toda potential. The difference is in the construction of the loss function. For one of them, the log-cosh error function is used for the energy loss. In the other result, the MSLE function is used. We note that the potential function in the two results is constructed by the

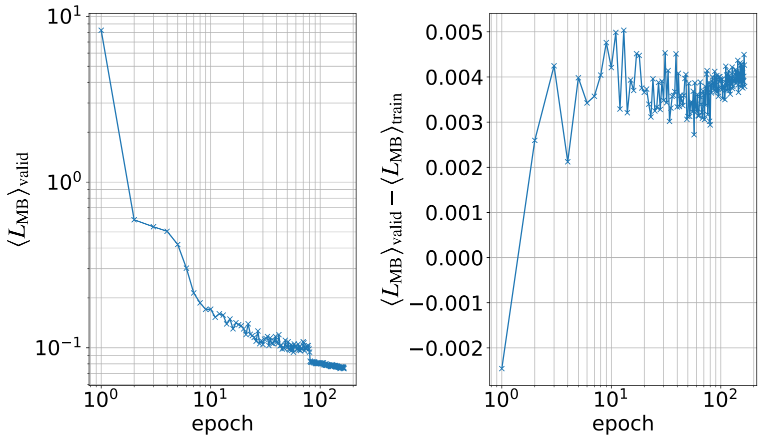


Figure 4.17: Change of the loss in the first example. The neural network is trained for 165 epochs. The learning rate is reduced at epoch 79 and reset the weights in the Adam optimizer. The validation loss almost converges.

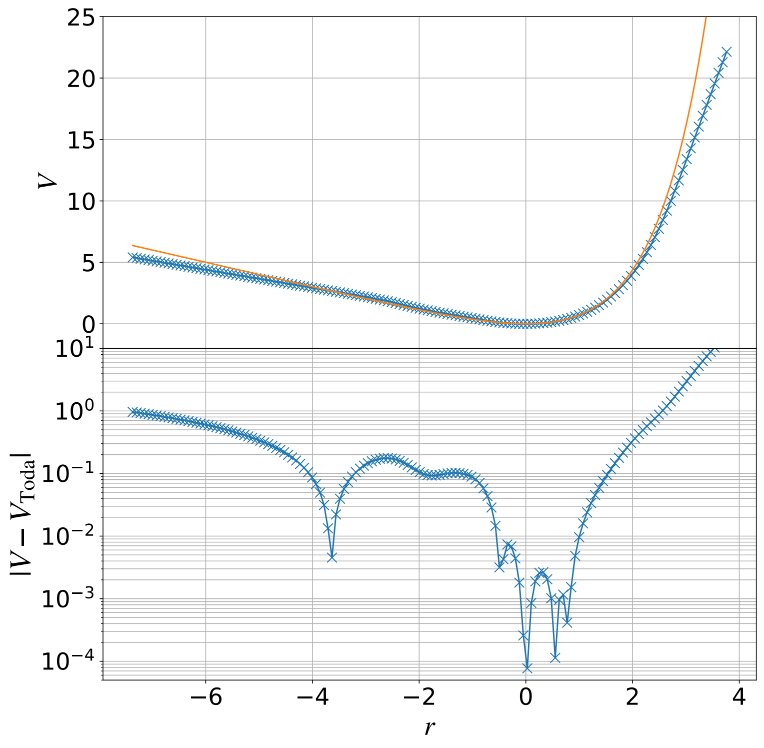


Figure 4.18: Obtained potential function in the first example. The upper figure shows the trained potential function and the true function. The crosses show the trained potential function after removing the constant and the linear function. The orang line is the true Toda potential whose parameters are $J = 1$ and $\alpha = 1$. The lower figure shows the relative difference between the neural potential function and the true Toda potential.

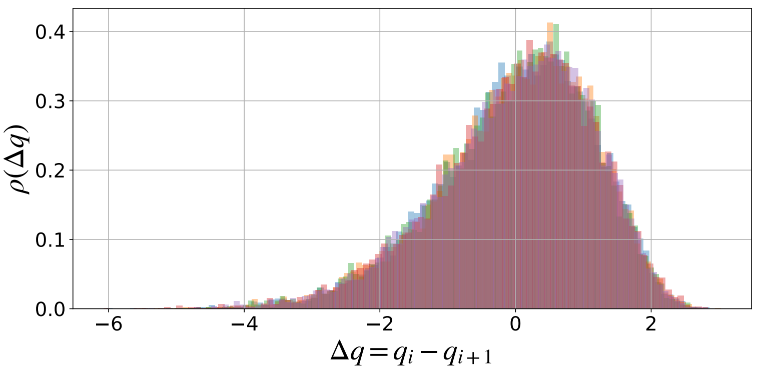


Figure 4.19: Distribution of $\Delta q = q_i - q_{i+1}$ of the Boltzmann distribution of the Toda lattice at $T = 2$. The data is produced from the Boltzmann distribution by using the HMC. Although there are four plots with different colors, they are overlapping as the shapes are the same. The distribution is located in the area $-4 < \Delta q < 2$. The number of samples is 10^4, and they are sampled from the Boltzmann distribution by HMC. The temperature is 2.

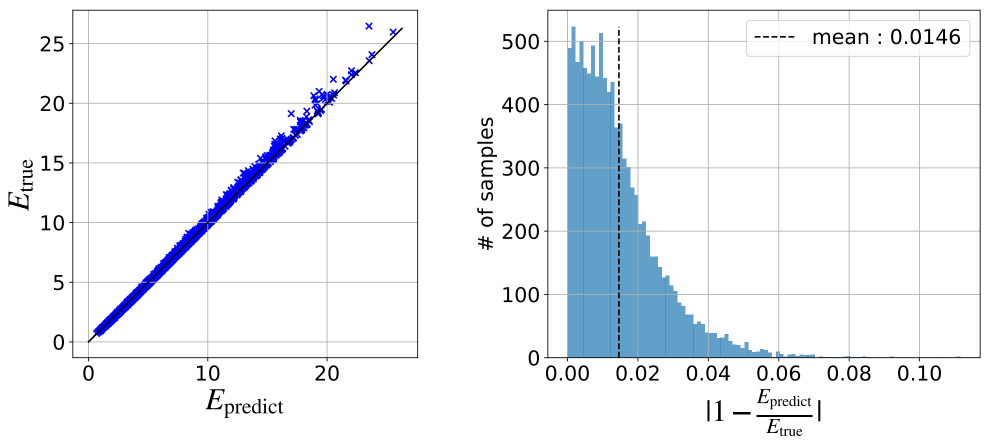


Figure 4.20: Energy produced by the trained potential function and the true Toda potential function. The left figure shows the correspondence between the predicted energy and the true energy. The right figure shows the distribution of the relative error of the energy. The number of samples is 10^4, and they are sampled from the Boltzmann distribution by HMC. The temperature is 2.

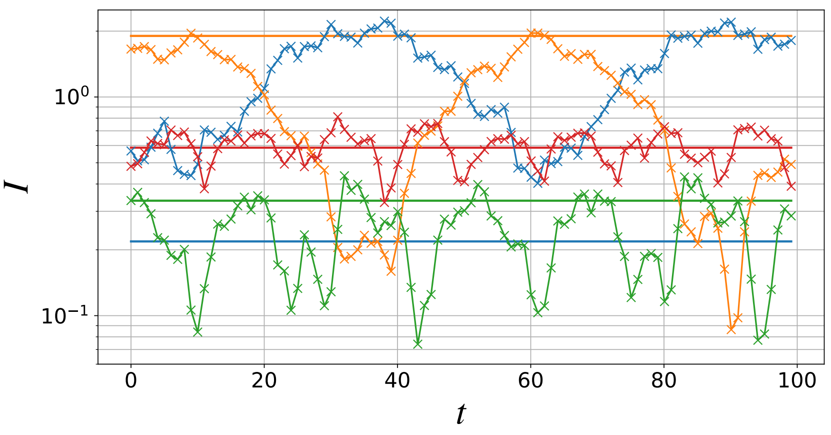


Figure 4.21: Time series of action variables predicted by the trained neural networks. The colors correspond to the index of action variables. The relations are as follows: I_1 (blue), I_2 (orange), I_3 (green), I_4 (red). The lines plotted by crosses are the data predicted by the neural networks. The straight lines are the values calculated by Eq. (2.43). The color correspondence is the same as the predictions by the neural network. We omit the plot of I_0, which corresponds to the momentum of the center of motion because it is not a prediction. The energy of the initial state is $E = 5.69$. The conservation score η_C is 0.357.

residual neural networks without additive bias. It means that the input variables are just multiplied by the weights. The final result where the neural network fails to find the Toda potential but reaches a lower loss value. The neural network does not find the Toda lattice but finds a harmonic oscillator potential function.

We show the first two results. Fig. 4.17 shows the loss change of the first result. Since the improvement of learning seems to be small at 79 epochs, we reduce the learning rate from 10^{-3} to 10^{-4} and reset the memory of the Adam optimizer. The learning process does not converge completely, but the improvement becomes too small. Thus, we stop learning and observe the neural network. The trained potential function is shown in Fig. 4.18. The neural network extracts the true potential function. The error of the prediction increases at the points far from zero. The region is determined by the distribution of the training data. Since the training data is sampled from the Boltzmann distribution at $T = 2$, the difference $\Delta q = q_i - q_{i+1}$ is sampled from limited ranges. Fig. 4.19 shows the distribution of Δq at $T = 2$. As shown in the figure, the number of data with $\Delta q < -4$ or $\Delta q > 2$ is small. Thus, the neural network learns the true potential function in the area bounded by the distribution. In fact, in Fig. 4.19, the difference of the prediction from the true potential increases when r is out from the area. Thus, the neural network success in learning the Toda potential function. It is also confirmed by the relative error of the prediction from the true energy. Fig. 4.20 shows the accuracy of the energy prediction by the neural network. From the plot, the relative error of the prediction is 1.46%. Although this result is worse than the previous simple task, the error can be small. In this sense, we conclude that the extraction of the potential function is achieved. On the other hands, as shown in Fig. 4.21, the learning the canonical transformation fails. In fact, the conservation score η_C is 0.357.

The reason for the failure seems to be the ratio of each part in the loss function. It is expected that the energy loss affects learning too strongly to fail the canonical translation learning. One of the most straightforward approaches to improve the behavior is changing the coefficients of the parts in the loss function. In this approach, however, the importance of the part of the loss function is expected to be uniformly decreased. Since the energy loss gives directions to the desirable integrable systems, it is suitable to avoid decreasing the effects.

From the viewpoints, we change the error function of the energy loss and compare the performance. So far, we use the log-cosh error function for energy loss. As shown in Fig. 4.3, the log-cosh error function increases faster than MSLE and MSRE. Thus, we consider MSLE and MSRE for energy loss.

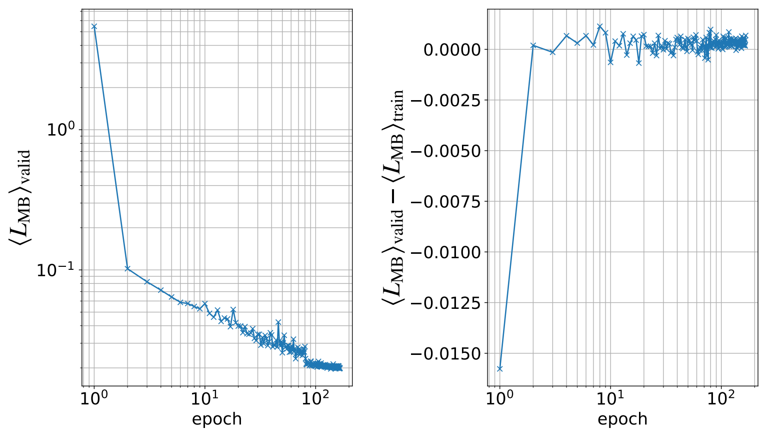


Figure 4.22: Change of the loss in the second example. The neural network is trained for 166 epochs. The learning rate is reduced at epoch 80, and reset the weights in the Adam optimizer. The validation loss almost converges.

According to the results of the previous task shown in Table. 4.2.5, MSLE is the second best. From the observation, we replace the error function with MSLE and compare the performance. The total loss function is given by

$$\begin{aligned} L &= L_I + L_E + L_p, \\ L_I &= \frac{1}{N N_{\mathrm{time}}} \sum_{i=1}^{N} \sum_{k=1}^{N_{\mathrm{time}}} \left\langle |\log(I_i(t_0)+1) - \log(I_i(t_k)+1)|^2 \right\rangle_D , \\ L_E &= \left\langle |\log(H(x)+1) - \log(E_{\mathrm{exact}}+1)|^2 \right\rangle_D , \\ L_p &= \frac{1}{N} \sum_{i=1}^{N} \left\{ \log\cosh(\langle p_i \rangle_D) + \log\cosh(\langle p_i^2 \rangle_D - \langle p_i^2 \rangle_{\mathrm{thermal}}) \right\} , \end{aligned} \tag{4.23}$$

Fig. 4.22 shows the loss change of the learning process using the replaced loss function. Since the improvement of learning seems to be small at 80 epochs, we reduce the learning rate from 10^{-3} to 10^{-4} and reset the memory of the Adam optimizer. The trained potential function is shown in Fig. 4.23. The neural network extracts the true potential function, but the error is larger than the previous one. The tendency of the difference is the same as the previous result. Fig. 4.24 shows the accuracy of the energy predicted by the neural network. From the plot, the relative error of the prediction is 2.46%. Compared with the performance of the previous result shown in Fig. 4.20, the error of prediction is large. Moreover, the error of the prediction of the higher energy samples is larger than the prediction of the small energy. The observation is natural because the MSLE evaluates the relative error. Whereas, as shown in Fig. 4.25, the trained canonical transformation is improved. η_C is 0.145, and it is less than η_C of the previous one. Thus, the replacement of the error function makes the learning process improved. Since the trained canonical transformation and the trained potential function are sufficiently accurate, it is expected that η_{NN} is low. Fig. 4.26 shows the action variables obtained by the trained canonical transformation and trained potential function. η_{NN} is 0.238 and lower than the previous score, $\eta_{\mathrm{NN}} = 0.432$. Consequently, we obtain the Toda potential by the neural network with the loss function defined by Eq. (4.23).

Next, we show the result where the neural network has an additive bias. The result elucidated below is representative of all the results where the neural networks have the additive bias. Concretely, the neural network structure for the potential function is constructed by the residual neural network with the additive bias. We note that the loss function used in the result is defined by Eq. (4.23). Excepting the performance of the trained canonical transformation, the same behavior is observed even if the loss function is defined by the previous one described in Eq. (4.22).

Fig. 4.27 shows the loss change of the learning process using the replaced loss function. Since the

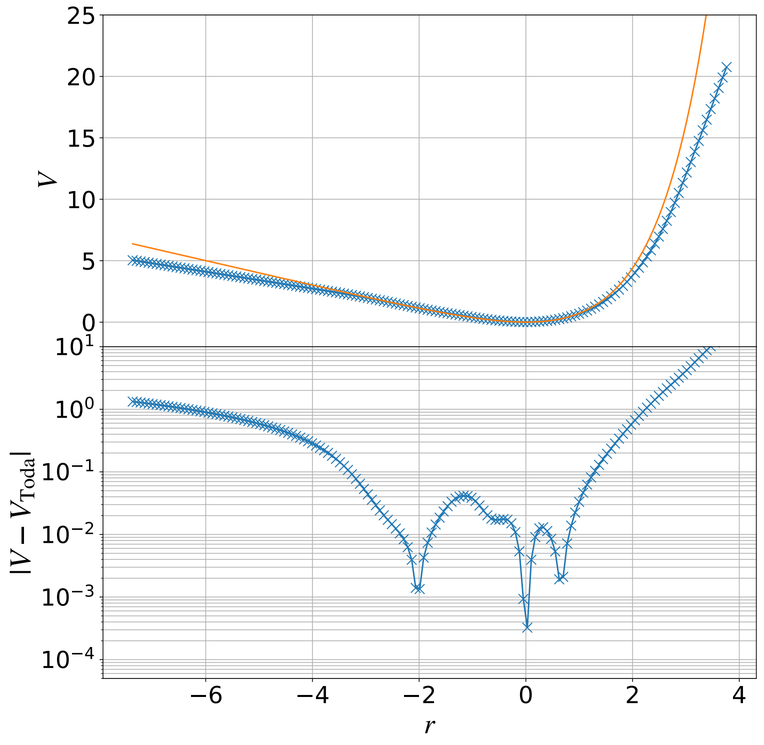


Figure 4.23: Obtained potential function in the second example. The upper figure shows the trained potential function and the true function. The crosses show the trained potential function after removing the constant and the linear function. The orang line is the true Toda potential with $J = 1$ and $\alpha = 1$. The lower figure shows the relative difference between the neural network and the true Toda potential function.

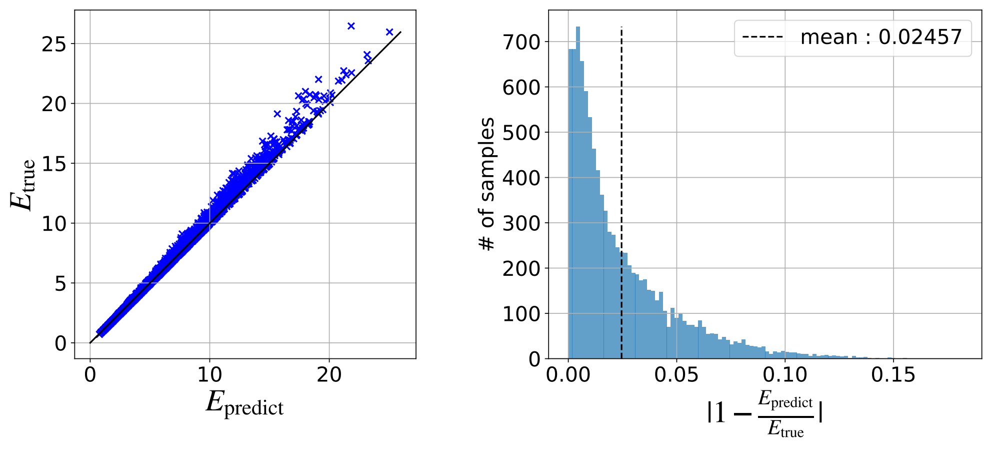


Figure 4.24: Energy predictions by the trained potential function compering with the true Toda potential function. The left figure shows the correspondence between the predicted energy and the true energy. The right figure shows the distribution of the relative error of energy. The number of samples is 10^4, and they are sampled from the Boltzmann distribution by HMC.

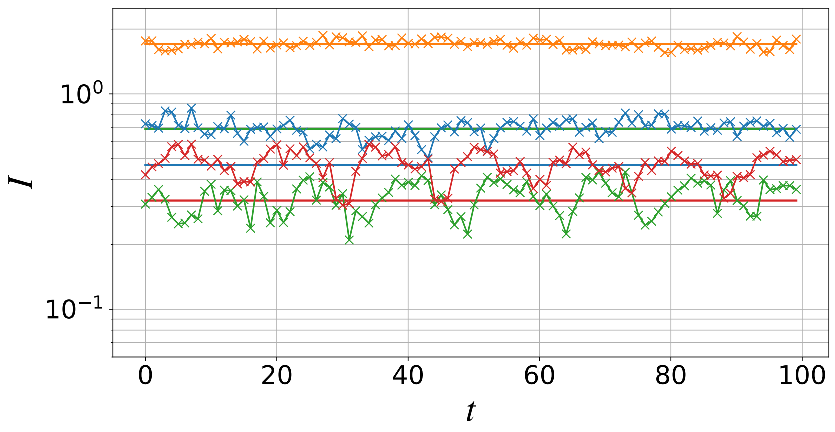


Figure 4.25: Time series of action variables predicted by the trained neural networks. The colors correspond to the index of action variables. The relations are as follows: I_1 (blue), I_2 (orange), I_3 (green), I_4 (red). The lines plotted by crosses are predicted by the neural networks. The solid lines are the values calculated by Eq. (2.43). The color correspondence is the same as the predictions by the neural network. We omit the plot of I_0, which corresponds to the momentum of the center of motion because it is not a prediction. The energy of the initial state is $E = 5.69$. The conservation score η_C is 0.145.

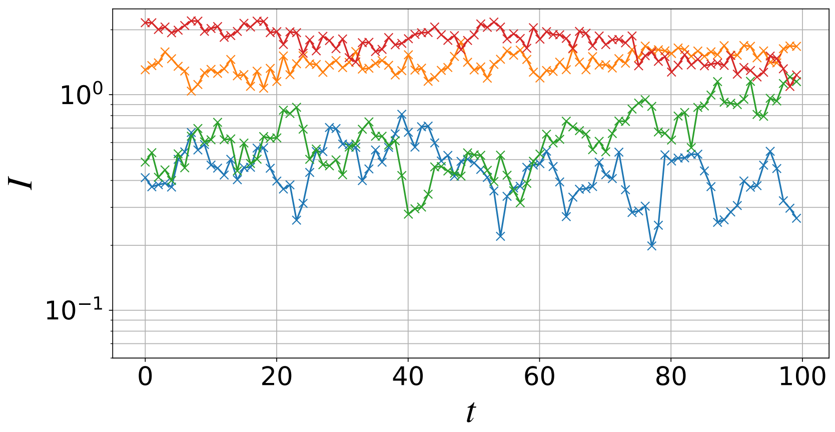


Figure 4.26: Time series of action variables predicted by the trained neural networks with trajectories generated by the trained potential function. The colors correspond to the index of action variables. The relations are as follows: I_1 (blue), I_2 (orange), I_3 (green), I_4 (red). The lines plotted by crosses are predicted by the neural networks. We omit the plot of I_0, which corresponds to the momentum of the center of motion because it is not a prediction. The conservation score η_{NN} is 0.238.

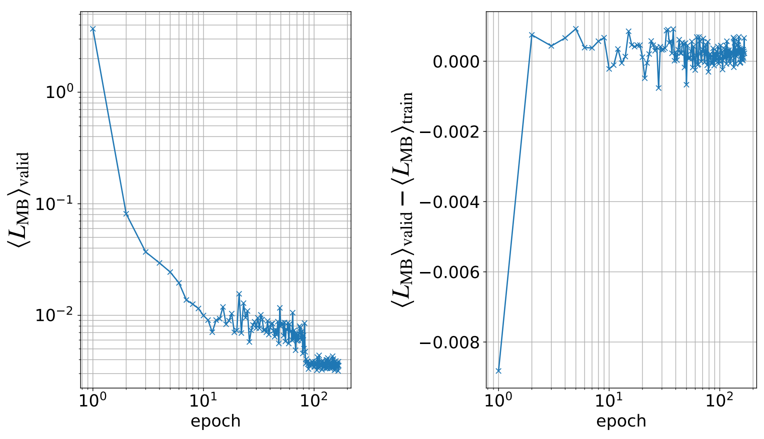


Figure 4.27: Change of the loss of the potential function constructed by the residual neural network with the additive bias. The neural network is trained for 167 epochs. The learning rate is reduced at epoch 82 and reset the weights in the Adam optimizer.

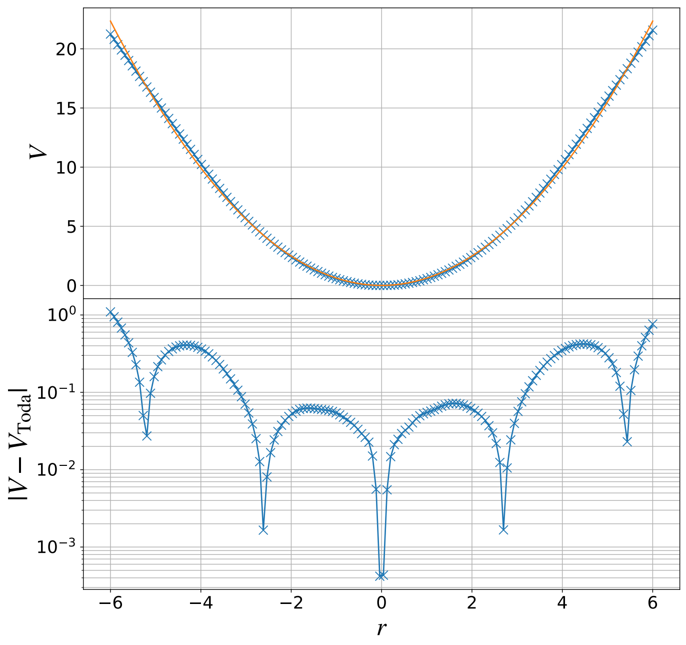


Figure 4.28: Obtained potential function constructed by the residual neural network with the additive bias. The upper figure shows the trained potential function and the true function. The plots with crosses show the trained potential function after removing the constant and the linear function. The orange line is the fitting harmonic oscillator potential ax^2. The coupling constant is estimated to be $a = 0.63$. The lower figure shows the relative difference between the neural network and the fitting function.

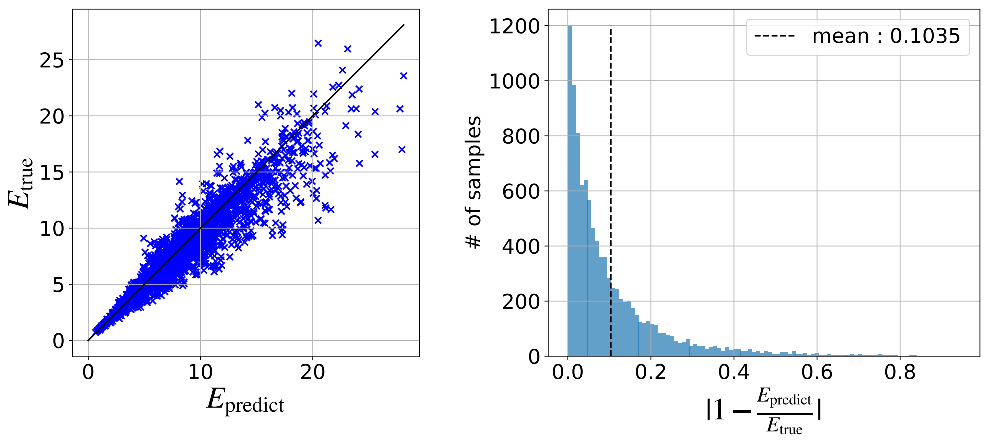


Figure 4.29: Energy prediction by the trained potential function comparing with the input energy data. The left figure shows the correspondence between the predicted energy and the true energy. The right figure shows the distribution of the relative error of energy. The number of samples is 10^4, and they are sampled from the Boltzmann distribution by HMC.

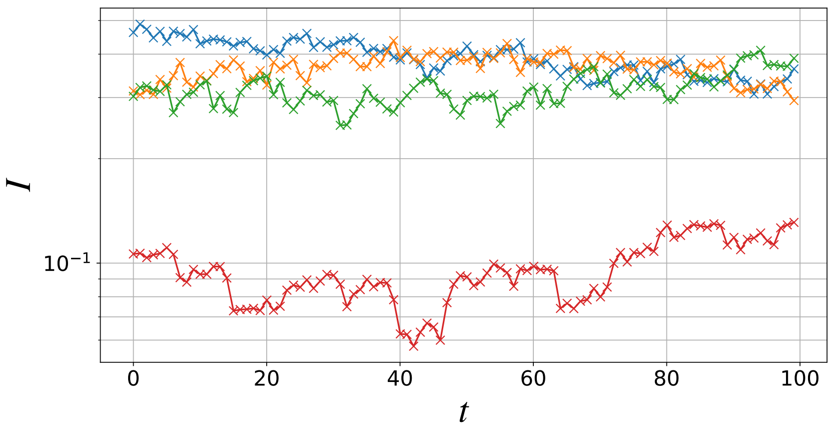


Figure 4.30: Time series of action variables predicted by the trained neural networks with trajectories generated by the trained potential function. The colors correspond to the index of action variables. The relations are as follows: I_1 (blue), I_2 (orange), I_3 (green), I_4 (red). The lines plotted with crosses are predicted by the neural networks. We omit the plot of I_0, which corresponds to the momentum of the center of motion because it is not a prediction. The conservation score η_{NN} is 0.159.

improvement of learning seems to be small at 82 epochs, we reduce the learning rate from 10^{-3} to 10^{-4} and reset the memory of the Adam optimizer. The trained potential function is shown in Fig. 4.28. The neural network does not learn the Toda potential function, but the harmonic oscillator potential function. The behavior causes the penalty for energy loss because the energy difference increases. The intuition is valid according to Fig. 4.20. From the plot, the relative error of the prediction is 10.3%. Compare the performance of the previous result shown in Fig. 4.24. Whereas, as shown in Fig. 4.21, the action variables are conserved. In fact, η_{NN} is 0.159. The score is less than the previous one, which is 0.238. Therefore, the neural networks take advantage of the conservation at the expense of energy loss.

The reason why the above results are observed is that the Toda lattice is approximately equivalent to the harmonic oscillator chain at low temperatures. As shown by Eq. (2.45) and Eq. (2.34), $K(I)$ of the Toda lattice is approximated by $K(I)$ of the harmonic oscillator. If the temperature is low, there is a small difference between the two systems. Thus, if the temperature of the Boltzmann distribution is sufficiently high, the learning process is expected to find the Toda potential even if the layers in the potential function does not include the additive bias. Although the higher temperature provides a larger energy difference, the higher temperature also spread the data points. Thus, larger neural networks are required for the canonical transformation. Furthermore, states with higher energy require a shorter time interval for time developments. The cost of the integration of the time development increases in proportional to the number of time steps. Due to the above requirements, the simulations with the conditions have high numerical costs and are difficult to be performed. The calculation is left for future work.

From the viewpoint of the above discussion, it is wondered why the residual neural networks without the additive bias find the Toda potential function. We observe the behavior which is expected to cause the difference in the loss value. The reached loss value of the residual neural network without the additive bias is 0.0199. For the other case, the reached loss value is 0.00353. Thus, the former case can be regarded as the local minimum. There are two interpretations. One of them is that the former neural networks do not have sufficient generalization ability and are stuck to the local minimum. The properties, however, are not observed in the simple task where the input data are the points in the real space. It is one of the future works to reveal the reason why the residual neural networks without additive bias show the behavior. The other interpretation is that the latter neural network exhibits the overfitting. The number of parameters in the latter neural networks is larger than in the former case. Thus, it is difficult to deny that the neural network overfits the data.

Although several points to be revealed are left, we conclude that our aim is achieved. The aim is to show the method using the loss function defined by Eq. (4.6) find the integrable system consistent with the given Hamiltonian $K(I)$. The first two results show the learning method find the potential function close to the Toda lattice. In the second result, the method does not find the Toda potential function but finds another consistent function: the harmonic oscillator potential function. In this sense, the results show that our method works for the Toda lattice case at least.

4.3 Attempts to Explore Integrable Systems

In this section, we attempt to explore the integrable systems by the aforementioned approach. According to our strategy, we first make the Hamiltonian $K(I)$ described by the action variables. There are some constructions of $K(I)$. Here, we try to find the integrable system characterized by the following $K(I)$.

$$K(I) = I_0 + \sum_{i=1}^{N-1} \frac{I_i^2}{4}. \tag{4.24}$$

The reason why we consider this Hamiltonian is that their Boltzmann distributions are easily sampled by the Gaussian distribution. Furthermore, the Hamiltonian is not approximated by the harmonic oscillator, unlike the Toda lattice. Therefore, the neural networks are expected not to be stuck to the minimum corresponding to the harmonic oscillator. As far as we know, the integrable system whose Hamilton is described by Eq. (4.24) has not been found. We apply the training process the three types of potential

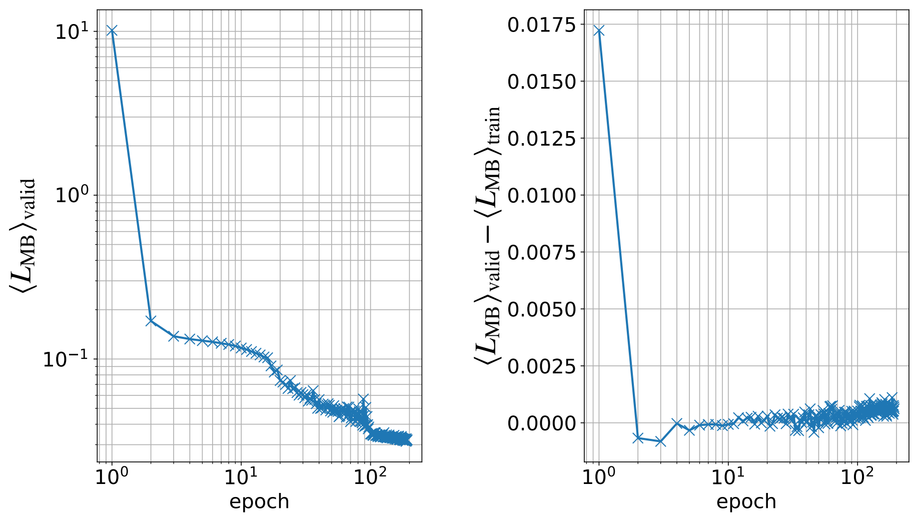


Figure 4.31: Change of the loss of training the potential function $V_{\mathrm{multi-body}}(q)$. The neural network is trained for 192 epochs. The learning rate is reduced at epoch 97 from 10^{-3} to 10^{-4} and reset the weights in the Adam optimizer. The left figure shows the difference between validation loss and training loss. The difference is small but increases slightly.

functions: $V_{\mathrm{chain}}(q), V_{\mathrm{fully}}(q)$, and $V_{\mathrm{multi-body}}(q)$ defined by Eq. (4.4). First, we show the result for $V_{\mathrm{multi-body}}(q)$. The neural networks assuming the multi-body interacting potential function exhibits the best performance in the three types. Next, we show the result for $V_{\mathrm{fully}}(q)$ and $V_{\mathrm{chain}}(q)$ to compare the behavior with the multi-body interaction case.

The data set is created by the Gaussian distribution equivalent to the Boltzmann distribution characterized by the Hamiltonian. The size of the validation data and the training data is 10^5. As well as the previous simulations, I_0 is fixed to be zero. It implies that the total momentum and the center of mass are always zero. For the neural networks of the canonical transformation, the number of the canonical transformation block is chosen to be 12. Here, the time for calculating the action variable loss is $t = 5$, and the number of time points is 5. The time interval is 0.05. We try to train several neural networks with different parameters. We use the softplus and SiLU function for the activation function. Moreover, the structure of the neural network is chosen to be the residual or dense neural network.

To begin with, we see the result for $V_{\mathrm{multi-body}}(q)$. We use the dense neural network for the potential function. The number of layers of the dense neural networks is 5, the dimension of the hidden layers is 128, and the activation function is SiLU. The loss change of the learning process is shown in Fig. 4.31. The difference between the validation loss and the training loss increases slightly. It implies the learning tends to be overfitting, but the value is small comparing the validation loss. Therefore, we stop learning and observe the trained neural network. As shown in Fig. 4.32, the action variables are conserved. The η_{NN} is 0.178 which is the best performance. Moreover, we evaluate the other two cases. The loss changes are shown in Fig. 4.33. The final loss values are larger than the multi-body interaction case. It indicates the worse performance in these two cases. The conservation scores are worse than the multi-body case. The time dependence of the action variables in the two case is shown in Fig. 4.34. η_{NN} of $V_{\mathrm{chain}}(q)$ is 0.597. Furthermore, the η_{NN} of $V_{\mathrm{fully}}(q)$ is 0.677. Therefore, we conclude that $V_{\mathrm{multi-body}}(q)$ is more appropriate for the desirable integrable system in the cases.

We see the detail of the obtained potential function. In the multi-body interaction case, the potential function has $\frac{N(N-1)}{2}$ input dimension:

$$V_{\mathrm{multi-body}}(q) = v(r_{12}, r_{23}, \ldots, r_{N-1N}),\ r_{ij} := q_i - q_j, \tag{4.25}$$

where N is the number of particles, and v is the potential function represented by the neural network. Here, as it is difficult to plot a 10-dimensional function, we plot the potential function by changing one variable and fixing a few variables. Moreover, we take two ways to plot the potential function. One of

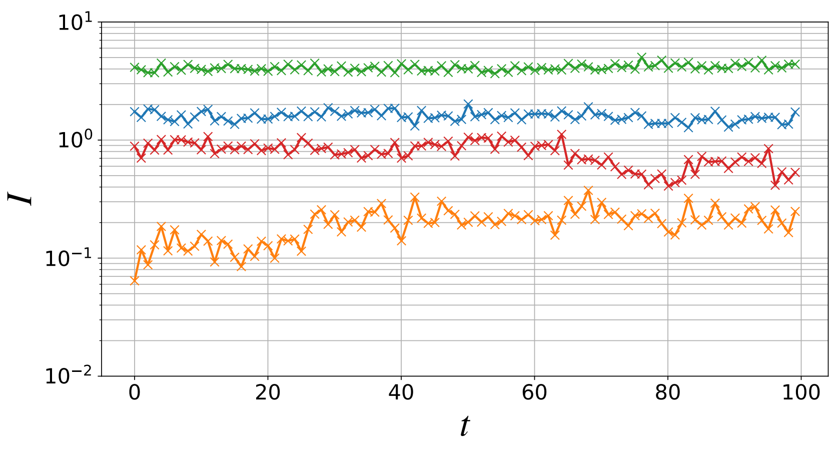


Figure 4.32: Time series of action variables predicted by the trained neural networks with trajectories generated by the trained potential function. The colors correspond to the index of action variables. The relations are as follows: I_1 (blue), I_2 (orange), I_3 (green), I_4 (red). We omit the plot of I_0, which corresponds to the momentum of the center of motion because it is not a prediction. The conservation score $\eta_{\rm NN}$ is 0.178.

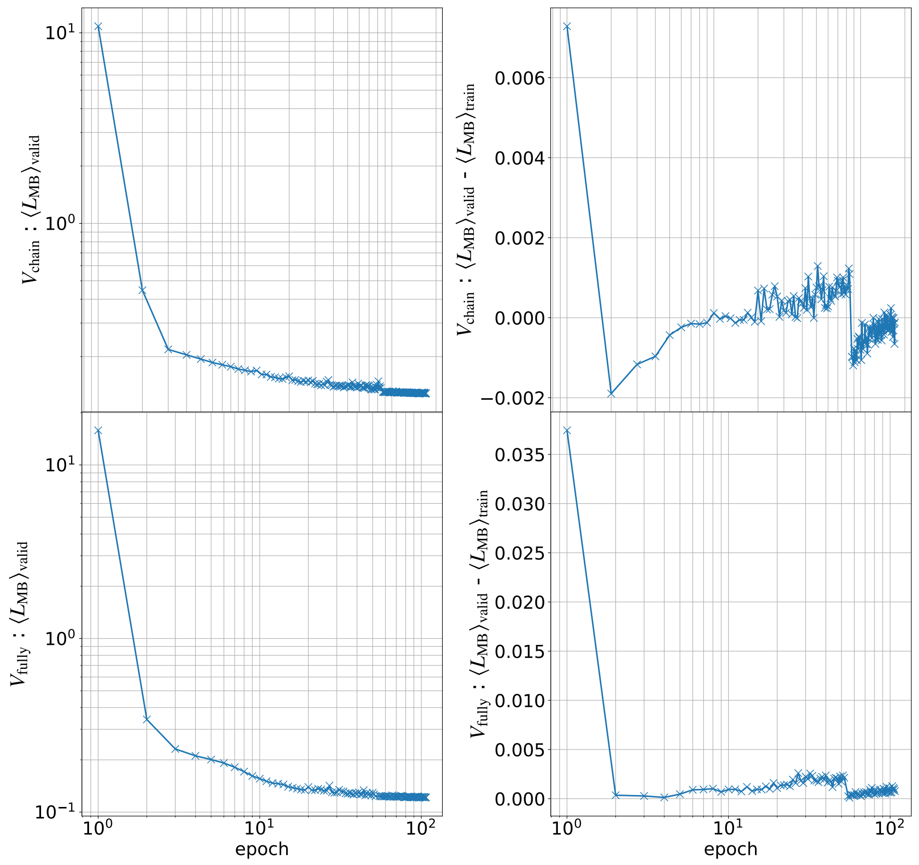


Figure 4.33: Change of the loss of training the potential function $V_{\rm chain}(q)$ and $V_{\rm fully}(q)$. The left figure shows the difference between the validation loss and the training loss. The difference is small but increases slightly.

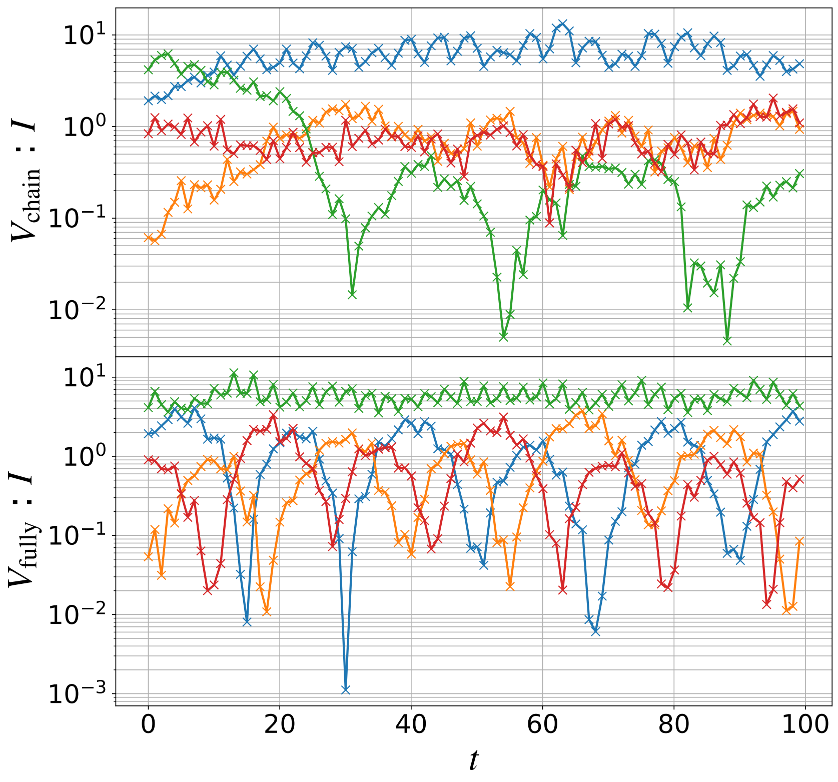


Figure 4.34: Time series of action variables predicted by the trained neural networks with trajectories generated by the two trained potential functions. The colors correspond to the index of action variables. The relations are as follows: I_1 (blue), I_2 (orange), I_3 (green), I_4 (red). The lines plotted with crosses are predicted by the neural networks. We omit the plot of I_0, which corresponds to the momentum of the center of motion because it is not a prediction. The conservation scores η_{NN} are $0.597(V_{\text{chain}}(q))$ and $0.677(V_{\text{fully}}(q))$ respectively.

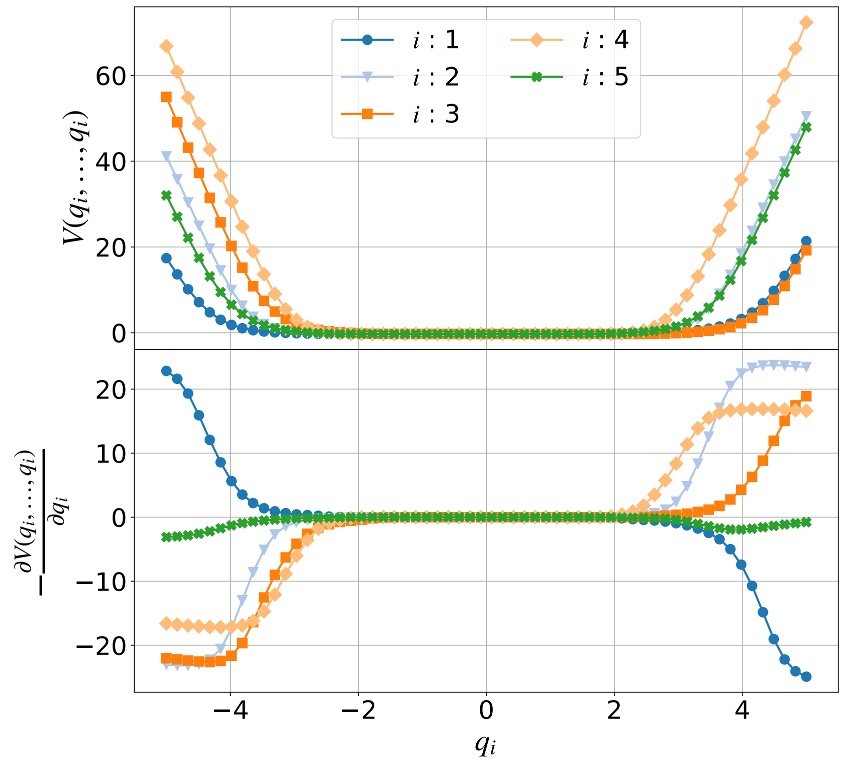


Figure 4.35: Obtained potential function $V_{\text{multi-body}}(q)$ represented by the neural network. The plot is obtained by changing the variable q_i and fixing the others to be zero. The lower figure shows the force obtained from the potential function.

them is to plot the potential function with a variable q_i, but the other variables are chosen to be zero. The second one is to plot the potential function with two variables. We check the dependence of v with the two variables q_i and q_j. The plots by the first method are shown in Fig. 4.35. The figure shows that the trained neural network does not have the same dependence on each variable. The potential function is constant in the support, $-2 < q_i < 2$. The plots by the second method are shown in Fig. 4.36. The figures show that the change in the potential function is parallel translation. It indicates that the potential function depends on the summation of the positions. For instance, in the case described by the upper left figure of Fig. 4.36, $V(x = (q_1, q_2, \mathbf{0}))$ is moved in parallel. Moreover, the shift of the potential function is different depending on the changed variable. It implies the potential function depends on the weighted summation of the positions. We also observe the same behavior by introducing the simple toy function defined by

$$V(q) = \left(\frac{3}{2}\sum_{i=1}^{N} q_i - 2\right)\left(\frac{3}{2}\sum_{i=1}^{N} q_i + 2\right)\left(\mathcal{H}\left(\sum_{i=1}^{N} q_i - 2\right) + \mathcal{H}\left(2 - \sum_{i=1}^{N} q_i\right)\right), \tag{4.26}$$

where $\mathcal{H}(x)$ is the Heaviside step function. A sketch of the graph of the simple toy function is shown in Fig. 4.37. It is obvious that the potential function depends on just the center of mass is an integrable system:

$$\begin{aligned} H &= \sum_{i=1}^{N} \frac{p_i^2}{2} + V\left(\sum_{i=1}^{N} q_i\right), \\ &= \sum_{k=1}^{N-1} \frac{P_k^2}{2} + \frac{P_0^2}{2} + V(Q_0), \end{aligned} \tag{4.27}$$

where $\{P_k\}$ are the variables transformed by the Hartley transformation, and Q_0 is the 0-th variable of the Hartley transformed position coordinate (see Eq. (2.31) and Eq. (4.8)). Q_0 is nothing but the center

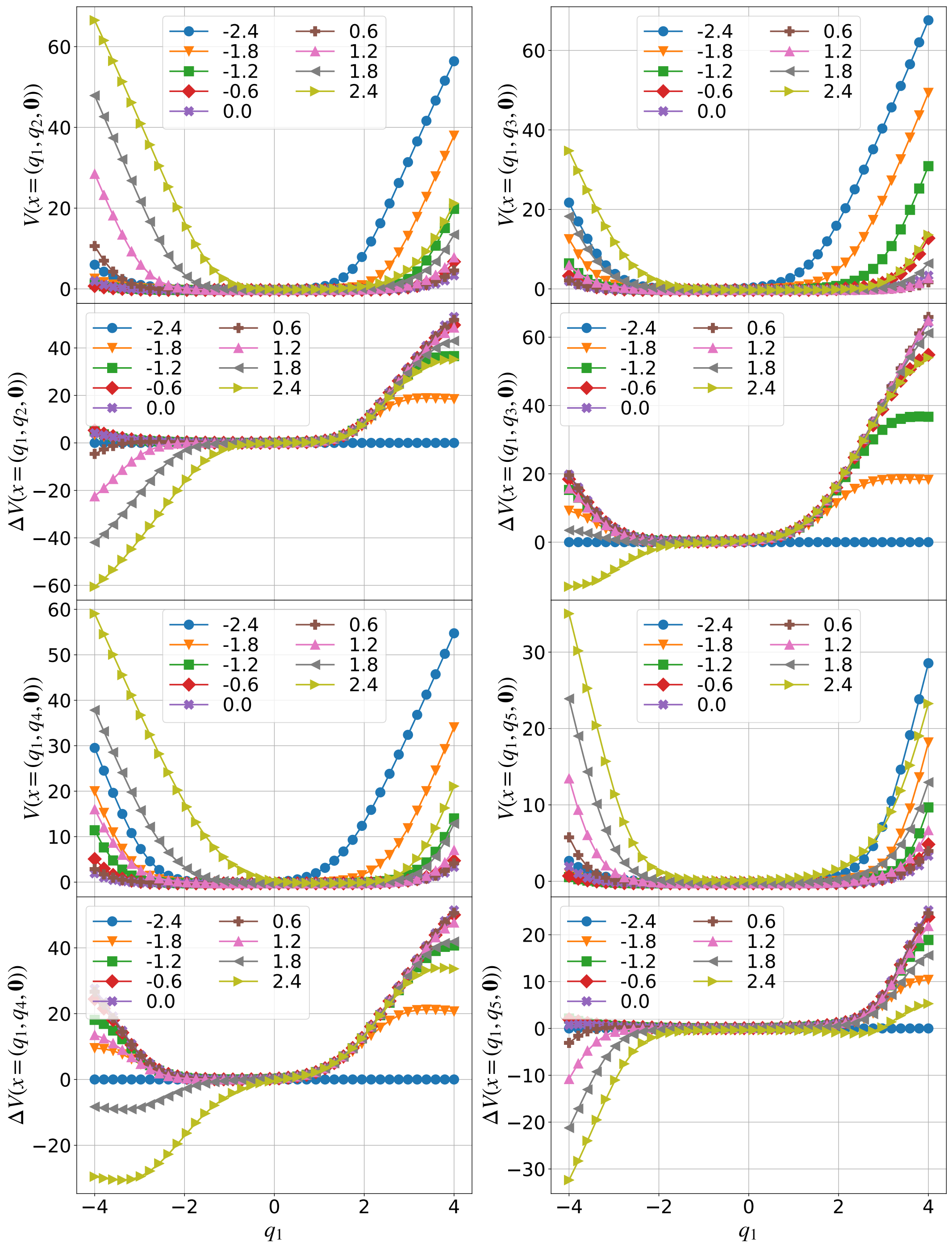


Figure 4.36: Obtained potential function $V_{\text{multi-body}}(q)$ represented by the neural network. The plot is obtained by changing the variable q_1 and fixing the others to be constant. In each figure, only one variable among $q_2, \ldots, q_5$ has a finite value. $x = (q_i, q_j, \mathbf{0})$ stands for the vector whose i-th element and j-th element have finite values but the oters are zero. The different color lines with crosses correspond to the various constants of the specific variable. ΔV stands for the difference of the potential function between the first finite constant and the others. For instence, in the upper left figure, $\Delta V(x = (q_1, q_2, \mathbf{0})) = V(x = (q_1, -2.5, \mathbf{0})) - V(x = (q_1, q_2, \mathbf{0}))$.

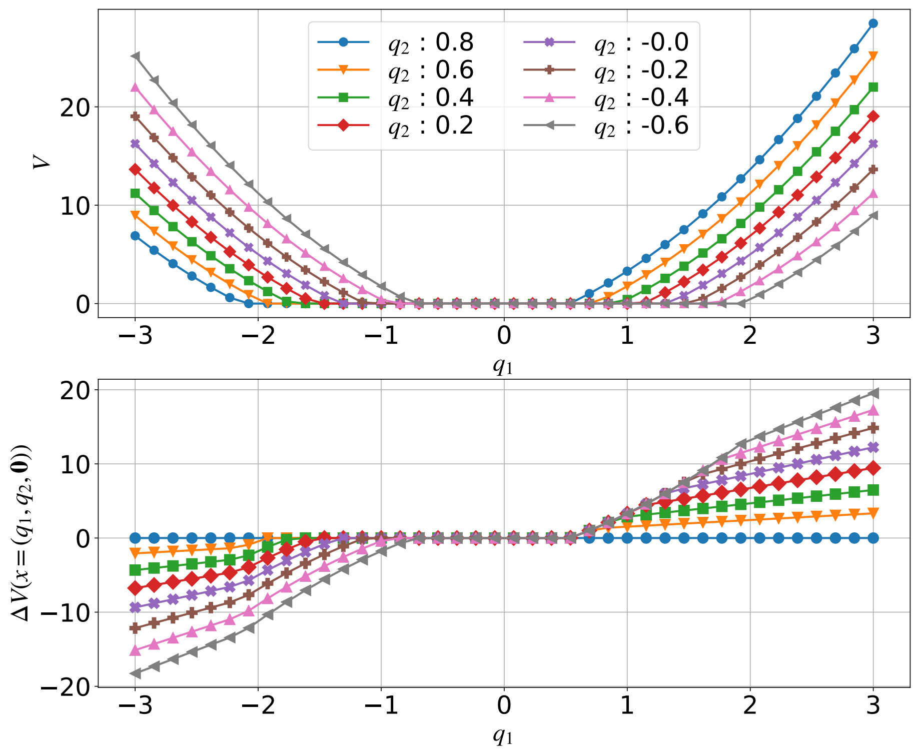


Figure 4.37: Simple example of the potential function showing the same behavior of the trained potential function. The function is defined in Eq. (4.26). The behavior is similar to one of the trained potential function shown in Fig. 4.36.

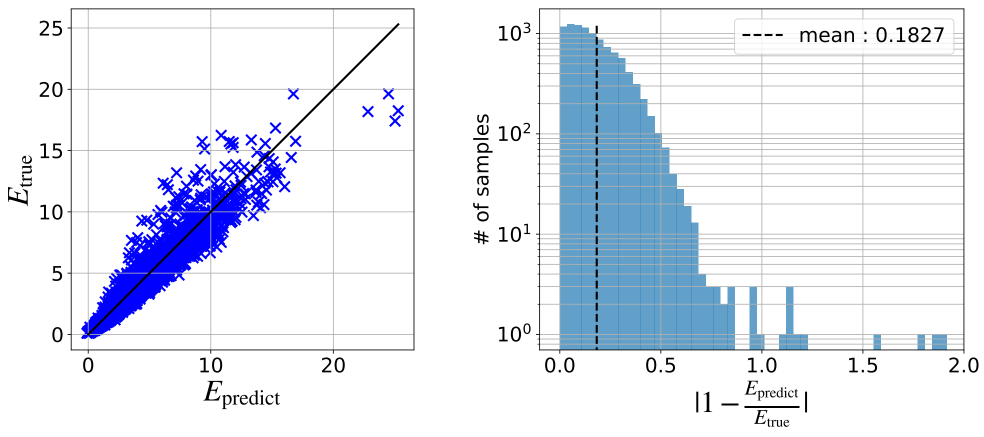


Figure 4.38: Energy predictions by the trained potential function compering the input energy data. The left figure shows the correspondence between the predicted energy and the true energy. The right figure shows the distribution of the relative error of energy. The number of samples is 10^4, and they are sampled from the Boltzmann distribution equivalent to the Gaussian distribution.

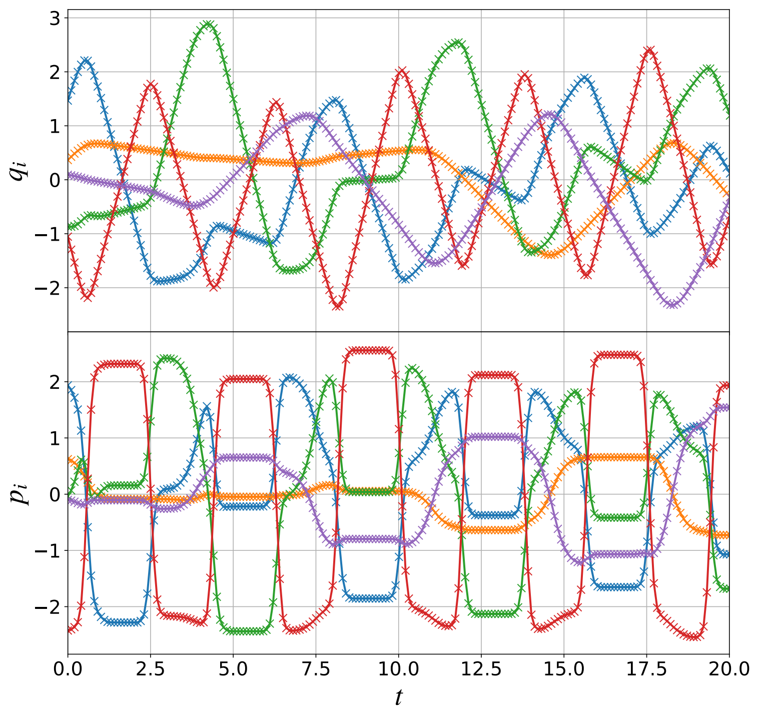


Figure 4.39: Time series of the coordinates in the real space generated by the trained potential function. The colors correspond to the site number. The upper figure shows the time series of the position q_i, and the lower figure shows the momentum. The relations are as follows: $i = 1$ (blue), $i = 2$ (orange), $i = 3$ (green), $i = 4$ (red), and $i = 5$ (magenda).

of mass, and the Hamiltonian is separable into the $N - 1$ free motions and the center of mass system. However, the motion driven by the Hamiltonian is not bounded, and $K(I)$ is not equal to our desirable Hamiltonian given by Eq. (4.24). In fact, Fig. 4.39 shows the system exhibits the bounded motion. Furthermore, the action variables are given by the radii of the latent space and the Eq. (4.27) is deformed to

$$H = \sum_{k=1}^{N-1} I_k + \mathscr{K}(I_0),\ I_k = \frac{P_k^2}{2},\ I_0 = \frac{1}{2\pi}\oint P_0 dQ_0, \tag{4.28}$$

where $\mathscr{K}$ is the function equivalent to $\frac{P_0^2}{2} + V(Q_0)$. From the point of view, one of the candidates of the trained potential function is characterized by

$$H = \sum_{k=1}^{N-1}\left(\frac{P_k^2}{2} + V_k(Q_k)\right) + \frac{P_0^2}{2} + V_{\mathrm{CM}}(Q_0), \tag{4.29}$$

where V_k and V_{CM} are defined as the functions satisfying the following condition:

$$\begin{aligned} \frac{P_k^2}{2} + V_k(Q_k) &= \frac{I_k^2}{4},\ I_k = \frac{1}{2\pi}\oint P_k dQ_k, \\ \frac{P_0^2}{2} + V_{\mathrm{CM}}(Q_0) &= I_0,\ I_0 = \frac{1}{2\pi}\oint P_0 dQ_0. \end{aligned} \tag{4.30}$$

Through the similar calculation in Eq. (2.36), we obtain the potential function as the follows. For simplicity, we focus on the k-th normal mode. From Eq. (4.30), we obtain the following relation:

$$I_k = \frac{1}{\pi\sqrt{2}}\int_{Q_k^-}^{Q_k^+}\sqrt{I_k^2 - 4V_k(Q_k)}dQ_k,\ I_k^2 - 4V_k(Q_k^{\pm}) = 0, \tag{4.31}$$

The simplest form of the potential function satisfying the equation is the function defined on the region $-\frac{\pi}{\sqrt{2}} < Q_k < \frac{\pi}{\sqrt{2}}$. It is nothing but a square-well potential. In fact, if the potential function is the infinite well potential, we can deform the equation as

$$\begin{aligned} I_k &= \frac{1}{\pi\sqrt{2}} \int_{Q_k^-}^{Q_k^+} \sqrt{I_k^2 - 4V_k(Q_k)} dQ_k, \\ &= \frac{1}{\pi\sqrt{2}} \int_{-\frac{\pi}{\sqrt{2}}}^{\frac{\pi}{\sqrt{2}}} I_k dQ_k, \\ &= I_k \end{aligned} \tag{4.32}$$

Thus, the obtained potential function by the neural network is perturbed square-well potential. We note that the action-angle variables are derived from the given potential function is easier than the opposite case. In fact, we find the text book where the above observations are introduced [143]. Therefore, we reveal that our approach can suggest to us the candidates of the integrable systems.

4.4 Conclusion

Through some experiments, we observe that the neural network approach successfully finds some integrable systems. The method extracts the Toda lattice and the integrable system characterized by a reduced Hamiltonian with perturbations. The trained neural networks provide insights for the integrable system, and we come up with the specific property of the potential function. Finally, we obtain the analytical representation of the suggested potential function. On the other hand, we cannot argue that the method tells us the true integrable system. In the present method, the criterion to find the integrable system depends on the conservation score η_{NN}. As we see in the above experiments, we mention that the trained neural networks "probably" capture the integrable system because the conservation score is "relatively" small. In fact, for the final results about the system characterized by Eq. (4.24), we can imply the system seems to be integrable on the grounds of the smaller conservation score than the Toda lattice case. According to the results, we just conclude that the method using the neural networks with the loss function has the potential to suggest the candidates of the integrable system, but it still has uncertainty. The improvement of the problem is left for future works.

Chapter 5

Summary and Outlook

5.1 Summary

In the present thesis, we introduce an approach to explore classical integrable systems assisted by the neural networks.

In Chap. 2, we review the classical integrable systems and the canonical transformations. Especially, we elucidate the action-angle variables, which are the essential canonical coordinates to characterize the integrable systems. The action-angle variables are the coordinates transforming the Hamiltonian to a specific form denoted as $K(I)$. Since the transformed Hamiltonian depends on just the action variables, they are naturally conserved quantities. We also review some approaches to construct integrable systems. One is the heuristic approach, which is applied to find the exponential potential function, the so-called Toda lattice. The model exhibits the soliton, which is a solitary running wave. The other is the ansatz approach to construct the integrable systems introduced by Calogero and Moser. The construction is generalized to the geometrical and group theoretical approach. The approaches indicate that some integrable systems are regarded as projections of the higher dimensional motion.

In Chap. 3, we review the neural networks mainly used for our approach. We introduce the basic concepts of neural networks by using feedforward neural networks. Especially, the dense neural networks and the residual neural networks are explained. The dense neural networks are the most famous neural networks, which are constructed by fully-connected layers. The residual neural networks have "skip connections" to avoid vanishing of the gradient. Furthermore, we introduce the adjoint method for learning by time series. The adjoint method calculates the gradient of the integration of ordinary differential equation efficiently. We also review the recent applications of neural networks to classical mechanics. Especially, we introduce the canonical transformations represented by the normalizing flow. We call the represented canonical transformation "neural canonical transformation". The normalizing flow is the neural network constructed by invertible neural networks. In particular, we introduce the realNVP which is one of the most famous invertible neural networks. Furthermore, we review the learning of the dynamical systems by the neural networks.

After the review, in Chap. 4, we propose a method to find the integral systems assisted by the neural networks. First, we examine the strategies to explore the integrable system. Furthermore, we consider the structure of the neural canonical transformation and the neural potential function. As a consequence of the discussion, we represent the definition of integrable systems as the loss function described by Eq. (4.2). To realize the loss function, we consider the four types of the error function, MSE, log-cosh error function, MSLE, and MSRE. For the evaluation of the error functions, we introduce two tasks. The first task is to learn the canonical transformation of the action-angle variables of the Toda lattice from the true trajectories. The second task is to learn the potential function of the Toda lattice from the data of energy and the coordinates in the real space. Furthermore, we define the two evaluation scores η_C and η_E for the quantitative evaluation. The score η_C represents the error of the conservation of action variables predicted by the trained neural canonical transformation. We evaluate the performance of the error functions for the learning of the canonical transformation. Consequently, the MSLE function is most appropriate for the first task. The score η_E evaluates the error of the energy prediction by the neural networks. Using the score, we compare the performance of the error functions for the learning of potential functions. As a result, we conclude that the log-cosh error function is the best error function for the second task. We also introduce another score function η_{NN} for the validity of the trained neural networks. The score evaluates the conservation of the action variables entirely generated by the neural networks. For instance, the trajectories in the real space are generated by the trained neural potential function. Moreover, the action variables are predicted by the trained neural canonical transformation. In this sense, the score η_{NN} estimates the validity of the trained neural networks. After that, we introduce two loss functions for the method to explore the integrable systems. The first loss function consists of just the loss of conservation of the action variables described by Eq. (4.21). By using the loss function, we extract some simple systems, *e.g.* the harmonic oscillator. However, the loss function is difficult to be informed about our desirable systems. Based on the observation, the second loss function is constructed by conservation of the action variables, the energy's loss function, and the statistical properties, which are described by Eq. (4.6). For the loss function, we use the log-cosh error function for the energy loss and the MSLE for the loss of conservation of the action variable The whole loss function is defend by Eq. (4.22). In other words, the loss function is constructed by the best error functions at each task. As shown in the results, however, the trained neural networks do not produce the well-conserved action variables. Therefore, we replace the log-cosh error function in the energy loss to the MSLE. The whole loss function is represented by Eq. (4.23). As a result, the loss function succeeds in capturing the Toda lattice. The results reveal that the loss function and the approach can extract integrable systems. After the above experiments, we apply the method to unknown integrable systems characterized by the transformed Hamiltonian defined by Eq. (4.24). The method indicates the integrable system whose potential function depends on the summation of the positions. Furthermore, the trajectories indicate the potential function is similar to the squared well potential. According to the insights, we determine the analytical representation of the suggested integrable system and prove that the potential function exhibits the transformed Hamiltonian. Concretely, the potential function is the superposition of the squared well potential of the modes obtained by the Hartley transformation. According to the results, we look for the reference and find the book describing the behavior of the potential function. Although the obtained potential function is not completely new, we conclude that our approach assisted by the neural network suggests to us the candidates of the integrable systems and helps us to explore the integrable systems.

5.2 Outlook

There remain several problems to be addressed in future studies. One of them is that our method in the present thesis restricts the search space of the Hamiltonian. There are two restrictions. The first one is that the action-angle variables are represented by the latent space according with [79]. Rigorously, it is not necessary to represent the action-variables through such axially space. Therefore, if we extend the search space to wider class of the integrable system, we represent the canonical transformation from the real space to the action-angle variables directly. The representation is, however, difficult because the angle variables have periodic conditions. Thus, they have large domain and the number of layers in the neural networks increases. If we represent the action-angle variables directly, we have to come up with methods to solve this problem.

The second restriction is that the approach requires us the concrete form of the transformed Hamiltonian $K(I)$ Our method looks for the potential function based on the given Hamiltonian $K(I)$. If the corresponded integrable system does not exist, the neural networks do not exhibit the performance. Furthermore, as we see in the final experiments, the simple $K(I)$ might produce a simple integrable system. To obtain the intriguing systems, the $K(I)$ might be complicated and should seem to exist but it is hard to become up with. One of the solutions is that the neural networks also learn the transformed Hamiltonian. In the method, we give the initial transformed Hamiltonian and the neural networks update it appropriately. The point to consider is how to generate the data set. In the present method, we generate the data set from the distribution defined by the $K(I)$. If we change the $K(I)$ on the learning process, the data set is resampled from the changed distribution. Therefore, we consider the loss change from the change of the distribution of the data set.

It is also a problem that the present method does not tell us the validity of the found integrable systems quantitatively. The introduced validation score η_{NN} evaluates the conservation and indicates the validity qualitatively. Therefore, it is difficult to estimate how the obtained potential function is similar to the true function. For the evaluation of the validity, one of the solutions is to introduce statistical inference techniques. For instance, in materials informatics, machine learning can estimate the prediction error [144, 145].

If the above problems are solved, our approach enables us to find a lot of integrable systems. As the Calogero-Moser systems show rich relations between the geometrical aspects of the Lie group and integrable systems, the found systems will provide new perspectives for the mathematics. On the other hands, as well as the Toda lattice, the found systems will exhibit new intriguing phenomena. If the phenomena are observed by experiments, we can understand the results by the found systems. Furthermore, our approach will open a new research field of the classical integrable systems. So far, the transformed Hamiltonian is not focused in the studies of the classical classical integrable systems. The reason is that the transformed Hamiltonian is difficult to be derived by hand. Our approach will enable us to reach the physics of the transformed Hamiltonian and the action-angle variables.

Here, we discuss future challenges of the third direction presented in Chap. 1. As mentioned in the chapter, the applications are not only tools to automate finding processes, but augment our searching ability. Furthermore, the concept of our approach is to find systems which have desired properties without experimental data. In this sense, our approach is close to the materials informatics. Our approach automates finding systems having the desired property which is the classical integrability. For extending our concept to other physical systems, we have to generalize our approach. In the present thesis, the essential point is that our approach needs desired properties and the forms of the systems. The properties are the classical integrability and forms of the Hamiltonian. As we show that the simple loss function finds the integrable systems in Sect. 4.2.6, we can obtain the systems satisfying the desired properties by unsupervised learning. Therefore, if we have powerful computers and enough memory storages, the neural networks extract the physical systems. Once we find the desired properties, the training data is simultaneously determined. For instance, as shown in Sect. 4.2.6 and Sect. 4.2.7, we make the training data after the discussion of the loss functions. Thus, if we define the desired properties and the forms of the systems in other fields, we can generalize our approach.

Machine learning methods automate the discovery process for us. They reduce trials and processes to find the systems. The thing that we should do is to define the desired properties and the forms of the systems. In other words, we have to determine directions of explorations. Furthermore, we have to interpret the results produced by the methods and understand the physical and mathematical concepts behind them. In the future, if we have such abilities and our approach is developed, we will obtain a lot of intriguing discoveries of physics assisted by machine learning.

Appendix A

Symplectic Integration of Adjoint Method

A.1 Higher-Order Symplectic Integrator

A.1.1 Symplectic integrator

In several cases, it is necessary to use numerical integrators for obtaining the evolution of systems. If the evolution is described by ordinary differential equations, the Runge-Kutta family are popular algorithms. They have versatility and, in addition, some of them enable controlling the accuracy automatically by adapting the time interval [146–148]. Because of the discretization of continuous variables such as time, they inevitably have the numerical error and there are more or less gaps from the analytical insights. For the Runge-Kutta family, excepting some partitioned schemes [149–151], the accumulated errors make the system drift from the constant energy surface even if the system is an autonomous system. If the system is described by the certain class of Hamiltonians, however, it is able to suppress the deviation by using symplectic integrators. Symplectic integrators are numerical integrators, which conserve the energy-like quantity if the system obeys the separable Hamiltonian $H = T + V$. One of the simplest examples is the symplectic Euler method for a harmonic oscillator system [152]. The difference equation is

$$\begin{aligned} q_{n+1} &= q_n + p_n h, \\ p_{n+1} &= p_n - q_{n+1} h, \end{aligned} \tag{A.1}$$

where h is the time interval. These iterative updates conserve not the energy but the following energy-like quantity[1]:

$$\begin{aligned} H_{pE} &:= \frac{\Gamma}{2}(q_n^2 + p_n^2 + p_n q_n h), \\ \Gamma &:= 1 + \frac{h^2}{6} + \frac{h^4}{30} + \cdots . \end{aligned} \tag{A.2}$$

In this sense, the symplectic integrator approximately conserves the energy during the time development. The essential point is that Eq. (A.1) is constructed from the two independent symplectic maps:

$$\begin{pmatrix} q_{n+1} \\ p_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -h & 1 \end{pmatrix} \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \begin{pmatrix} q_n \\ p_n \end{pmatrix}. \tag{A.3}$$

Each map conserves the volume of the phase space rigorously and whole update restricts the deviation.

Here, we assume the Hamilton is separable and the system is autonomous, $H = T(p) + V(q)$, where $T(p)$ is the kinetic term. By using the exponential map, the difference equation is written as

$$\begin{aligned} x_{n+1} &= e^{hD_V} e^{hD_T} x_n, \\ D_T &:= \frac{\partial T(p)}{\partial p}\frac{\partial}{\partial q}, \; D_V := -\frac{\partial V(q)}{\partial q}\frac{\partial}{\partial p} \end{aligned} \tag{A.4}$$

[1]To know the modified energy of the higher-order scheme, see [152].

where x_i is a vector (q_i, p_i), D_T and D_V are operators induced by the Poisson brackets with the kinetic term and the potential term in the Hamiltonian, respectively. Note that the exponential maps are rigorous solutions of the differential equations induced by the operators if the terms are time-independent. Through applying the Baker-Campbell-Hausdorff formula to Eq. (A.4), we obtain the"Hamiltonian" of the motion generated by the symplectic Euler equation[2]:

$$\begin{aligned} &e^{hD_V} e^{hD_T} = e^{hD_{H_{pE}}}, \\ &D_{H_{pE}} = D_H + \frac{h}{2}[D_V, D_T] + \frac{h^2}{12}\big([D_V, [D_V, D_T]] - [D_T, [D_V, D_T]]\big) + \cdots . \end{aligned} \tag{A.5}$$

From the correspondence between the definition of $D_{H_{pE}}$ and the expansion, we obtain the aforementioned energy-like quantity. In this sense, the symplectic integrators rigorously conserve the approximated energy.

A.1.2 Higher-order scheme

To obtain higher-order schemes, the derivation of symplectic integrators is generalized [132, 153–155]. The purpose is to improve the approximation of the Hamiltonian and it means to produce the more accurate modified Hamiltonian. As well as the Runge-Kutta family, multi-stage update reduces the deviation in symplectic integrators. The multi-stage update is represented by the exponential map as

$$x_{n+1} = \prod_{i=1}^{k} e^{hd_j D_V} e^{hc_j D_T} x_n \text{ or } \prod_{i=1}^{k} e^{hd_j D_T} e^{hc_j D_V} x_n, \tag{A.6}$$

where c_j and d_j are coefficients of each stage. Even if the integration is constructed by the multi-stage update, the each update is a symplectic map and whole update conserves the modified Hamiltonian. The coefficients are determined by the following procedure: expanding the r.h.s. of Eq. (A.6) and the exponential map of original Hamiltonian and make the coefficients equal to the original one up to the order h^k. Note that the coordinates are updated at different "time":

$$\begin{aligned} q_{i+1} &= q_i + hd_i \frac{\partial T}{\partial p}(p_i), \\ p_{i+1} &= p_i - hc_i \frac{\partial V}{\partial q}(q_{i+1}). \end{aligned} \tag{A.7}$$

Furthermore, it is obviously that the coefficients is independent from the order of acting the operators. Thus, when a combination of the coefficients is obtained, it is possible to construct two symplectic integrators at once.

A.1.3 Specific representation of fourth-order scheme

For the fourth-order scheme, two combinations of the coefficients are found [132, 153–155]:

$$\begin{aligned} &c_1 = c_4 = \frac{1}{2(2-2^{1/3})}, \; c_2 = c_3 = \frac{1-2^{1/3}}{2(2-2^{1/3})}, \\ &d_1 = d_3 = \frac{1}{2-2^{1/3}}, \; d_3 = -\frac{2^{1/3}}{2-2^{1/3}}, \; d_4 = 0, \end{aligned} \tag{A.8}$$

or,

$$\begin{aligned} &c_1 = 0, \; c_2 = c_4 = \frac{1}{2-2^{1/3}}, \; c_3 = -\frac{2^{1/3}}{2-2^{1/3}}, \\ &d_1 = d_4 = \frac{1}{2(2-2^{1/3})}, \; d_2 = d_3 = \frac{1-2^{1/3}}{2(2-2^{1/3})}. \end{aligned} \tag{A.9}$$

[2]To know the higher-order terms of the expansion, see [132].

For reducing the number of steps, let the kinetic term be constructed with just square of the momentum, which means $T \propto \frac{p^2}{2}$. Then, Yoshida fourth-order symplectic integrator is obtained with combining Eq. (A.8) and the second form in Eq. (A.6). In the symplectic partitioned Runge-Kutta family, there are also different higher-order schemes which satisfy the symplectic conditions and they are one of the generalization of the symplectic integrators [149–151].

A.2 Adjoint Method with Hamiltonian Dynamics

A.2.1 Hamiltonian of the adjoint method

In this section, we show the symplectic integrator is appraisal to the adjoint method. This argument is a corollary derived from the statements about the application of symplectic Runge-Kutta methods to the adjoint method in [82]. We assume the Hamiltonian is one of the certain class described as $H = \frac{p^2}{2} + V(q, \theta)$, where θ stands for the parameters characterizing the potential function. If the evolution of the system obeys the Hamiltonian, the dynamics of the adjoint variable is given as

$$\begin{aligned} \frac{d\lambda_i^p}{dt} &= \sum_j \frac{\partial^2 V}{\partial q_i q_j} \lambda_i^q, \\ \frac{d\lambda_i^q}{dt} &= -\lambda_i^p, \end{aligned} \tag{A.10}$$

where λ^p and λ^q are the former half of the adjoint variables and the latter half, respectively. At the first glance, the equation is separable and appropriate form for the symplectic integrator. The equation is, however, not closed because of the term in the r.h.s of the first formula in Eq. (A.10), which depends on the original coordinates. In this sense, it is not straightforward to show the validity. For elucidating the applicability of the symplectic integrators, we introduce the pseudo-Hamiltonian. Suppose the equation of the system and the adjoint equation induced the dynamics are given by Eq. (3.25), the pseudo-Hamiltonian is defined as

$$H^{ad} := \lambda^T f(x, \theta), \tag{A.11}$$

where the pair of λ and x is regarded as the conjugate variables. From the definition, it is easily to derive the equations by manipulations of the Hamiltonian formalism. Then, the whole evolution of the system is driven by the following exponential map as

$$\begin{aligned} \begin{pmatrix} x_{n+1} \\ \lambda_{n+1} \end{pmatrix} &= e^{hD_{H^{ad}}} \begin{pmatrix} x_n \\ \lambda_n \end{pmatrix}, \\ D_{H^{ad}} &= D_V^{ad} + D_T^{ad}, \\ D_T^{ad} &= p_i \frac{\partial}{\partial q_i} - \lambda_i^p \frac{\partial}{\partial \lambda_i^q}, \\ D_V^{ad} &= -\frac{\partial V}{\partial q_i} \frac{\partial}{\partial p_i} + \frac{\partial^2 V}{\partial q_i \partial q_j} \lambda_j^q \frac{\partial}{\partial \lambda_i^p} + \frac{\partial^2 V}{\partial q_i \partial \theta} \lambda_j^q \frac{\partial}{\partial \lambda_\theta}. \end{aligned} \tag{A.12}$$

Each operators in D_T^{ad} and D_V^{ad} are commutable. Due to the pseudo-Hamilton is not separable, the each exponential map in the expansion of Eq. (A.6) is not the symplectic. In this sense, the discretization with respect to the each Hamiltonian dynamics does not hold the symplectic properties, which means the error of the pseudo-Hamiltonian is accumulated by the time development. From the point of view of "the adjoint system of the difference equations", however, the application of the algorithm is validated [82]. The essential point is that the difference equation should conserve the inner product of the adjoint variables and the variations rather than the pseudo-Hamiltonian. Revisit the definition of the adjoint system. The adjoint system is induced by the variational equation of the Lagrangian functional including the loss function. Let L be a loss function depending on the final state $x(t_f)$ and $\mathcal{L}$ be the Lagrangian functional:

$$\mathcal{L} = L(x(t_f)) + \lambda(t_0)^T (x(t_0) - x_0) - \int_{t_0}^{t_f} \lambda^T(t)\left(\frac{dx}{dt} - f(x(t), \theta)\right)dt, \tag{A.13}$$

Let $\delta(t)$ be the variation of $x(t)$. We denote the initial variation $\delta(t_0)$ as δx_0. Then, performing functional derivative, the change of the functional by the small variation is

$$\delta\mathcal{L} = \left(\nabla_{x(t_f)} L\right)^T \delta(t_f) + \lambda(t_0)^T (\delta(t_0) - \delta x_0) - \int_{t_0}^{t_f} \lambda^T(t)\left(\frac{d\delta}{dt} - \partial_x f \delta(t)\right)dt. \tag{A.14}$$

The second term in the r.h.s is the initial condition of the variation. The third term in the r.h.s gives the dynamics to the variations. Thus, the first term is left and equal to the term in l.h.s:

$$\delta\mathcal{L} = \left(\nabla_{x(t_f)} L\right) \delta(t_f). \tag{A.15}$$

In other words, performing the partial integration on Eq. (A.14), we obtain another representation as

$$\delta\mathcal{L} = \left(\nabla_{x(t_f)} L - \lambda(t_f)\right)^T \delta(t_f) + \lambda(t_0)^T \delta x_0 - \int_{t_0}^{t_f} \left(\frac{d\lambda^T}{dt} - \lambda^T \partial_x f\right)\delta(t)dt. \tag{A.16}$$

In the same way, we obtain the final condition of λ and the dynamics of λ. Using the conditions, the l.h.s of Eq. (A.16) is given as

$$\delta\mathcal{L} = \lambda(t_0)^T \delta x_0. \tag{A.17}$$

The relation between Eq. (A.15) and Eq. (A.17) indicates that the conserved quantity. According with the correspondence, we rewrite the quantity $\left(\nabla_{x(t_f)} L\right)$ and δx_0 as $\lambda(t_f)$ and $\delta(t_0)$, respectively. Thus, through the variation of the functional is given as

$$\delta\mathcal{L} = \lambda(t_0)^T \delta(t_0) = \lambda(t_f)^T \delta(t_f). \tag{A.18}$$

It is nothing but the conserved quantity and the condition which is satisfied by λ and δ. More precisely, from the variational equations from the functional, the variations of x is obtained as

$$\begin{aligned}\frac{d\delta_i^q}{dt} &= \delta_i^p, \\ \frac{d\delta_i^p}{dt} &= -\sum_j \frac{\partial^2 V}{\partial q_i q_j}\delta_i^q,\end{aligned} \tag{A.19}$$

where δ_i^q is the upper half of the vector δ and δ_i^q is the lower half. It is easily confirmed that the inner product of (δ^q, δ^p) and (λ^p, λ^q) is conserved within the dynamics. Furthermore, one can proof that the inner product is conserved in the dynamics discretized by the symplectic integrators. In this sense, the discretization of the adjoint method by symplectic integrator is valid.

In another method, we can obtain the adjoint equation and the above relation from the chain rule of the time development. The chain rule of the time development is given by [81]

$$\begin{aligned}\frac{\partial L}{\partial x(t)} &= \frac{\partial L}{\partial x(t+\Delta t)}\frac{\partial x(t+\Delta t)}{\partial x(t)}, \\ \lambda(t) &= \lambda(t+\Delta t)\frac{\partial x(t+\Delta t)}{\partial x(t)}, \\ \frac{d\lambda(t)}{dt} &= \lim_{\Delta t\to 0}\frac{\lambda(t+\Delta t) - \lambda(t)}{\Delta t}, \\ &= \lim_{\Delta t\to 0}\frac{\lambda(t+\Delta t)}{\Delta t}\left(I - \frac{\partial x(t+\Delta t)}{\partial x(t)}\right), \\ &= \lim_{\Delta t\to 0}\left\{-\left(-\frac{\partial f(x)}{\partial x(t)}\right)^T \lambda(t+\Delta t) + O(\Delta t)\right\},\end{aligned} \tag{A.20}$$

where $\lambda(t+\Delta t)$ is equal to $\lambda(t)$ in $O(\Delta t)$. We proof the backpropagation of the symplectic integrator is equivalent to the discretization of the adjoint equation.

Appendix B

Deformations in Standard Integrable Systems

B.1 Decoupling Harmonic Oscillator Chain

In this appendix, we elucidate the ordinary method to decouple the harmonic oscillator chain. The well-known approach is started from the equation of motion with the ansatz. It is straightforward but the rigorous manipulation is unclear. The ansatz approach is supported by the constant-recursive sequence method for the diagonalization.

B.1.1 Standard approach

Usually, the decouple of the harmonic oscillator is started from the equation of motion. In the method, one use the ansatz that the solutions are supposed to be represented by time-periodic functions. The equation of motion of the harmonic oscillator chain is

$$\frac{d^2 q_i}{dt^2} = Jq_{i-1} - 2Jq_i + Jq_{i+1},\ i = 1, \cdots, N, \tag{B.1}$$

where J is the coupling constant and the coordinates satisfy the periodic boundary condition, $q_i = q_{N+i}$. Since the solutions are to be oscillatory, new variables are introduced by the discrete Fourier transformation:

$$\begin{aligned} Q_k &= \sum_{j=1}^{N} q_j \exp\left(\frac{2i\pi jk}{N}\right), \\ \frac{d^2 Q_k}{dt^2} &= -\omega_k^2 Q_k,\ \omega_k := \sqrt{4J}\sin\left(\frac{\pi k}{N}\right). \end{aligned} \tag{B.2}$$

Consequently, the harmonic oscillator chain is decoupled to N independent oscillators.

B.1.2 Constant-recursive sequence approach

To elucidate the aforementioned method, we focus on the Hamiltonian, which is sum of two quadratic form. The Hamiltonian of the harmonic oscillator chain is

$$H = \sum_{i=1}^{N} \frac{p_i^2}{2} + \frac{J}{2}(q_i - q_{i+1})^2 = \frac{1}{2}p^T p + \frac{J}{2}q^T L q,$$
$$L := \begin{pmatrix} 2 & -1 & & & -1 \\ -1 & 2 & & 0 & \\ & & \ddots & & \\ & 0 & & 2 & -1 \\ -1 & & & -1 & 2 \end{pmatrix}. \tag{B.3}$$

Decoupling the harmonic oscillator chain is nothing but the diagonalization of the matrix L. The eigenvector of L, denoted as ϕ, satisfies

$$\begin{aligned} &L\phi = \lambda\phi, \\ &-\phi(n-1) + (2-\lambda)\phi(n) - \phi(n+1) = 0, \ \phi(n) = \phi(n+N). \end{aligned} \tag{B.4}$$

The equation is able to be regarded as an order-2 homogeneous linear recurrence with constant coefficients [156]. As well as ordinary differential equations, the solution is represented by two independent solution, $\phi(n) = C_1\phi_1(n) + C_2\phi_2(n)$ and the solutions are obtained by the characteristic equation. The characteristic equation is constructed by substituting $\phi(n) = r^n$:

$$r^2 - 2\alpha r + 1 = 0, \ \alpha := \frac{2-\lambda}{2}. \tag{B.5}$$

Thus, the solution is

$$r = \alpha \pm \sqrt{\alpha^2 - 1}. \tag{B.6}$$

Although the sign of $\alpha - 1$ causes two cases, the periodic boundary condition of $\phi(n)$ fix the case that $\alpha < 1$. Hence, α can be represented as $\cos\theta$ and $\phi(n)$ is obtained as

$$\phi(n) = C_1 e^{+in\theta} + C_2 e^{-in\theta}, \tag{B.7}$$
$$\lambda_m = 2\big(1 - \cos(\theta)\big), \tag{B.8}$$

where $\theta = 2\pi m/N$. For convenience, we deform the representations as

$$\phi(n) = C\cos\big(kn + \delta\big), \ k := 2\pi m/N. \tag{B.9}$$

Normalization of the eigenvectors derive the relation of constants:

$$C = \sqrt{\frac{2}{N}}, \tag{B.10}$$
$$\delta = \pm\frac{\pi}{4}, \pm\frac{3\pi}{4}, \pm\frac{5\pi}{4}, \cdots \tag{B.11}$$

Here, we fix $\delta = -\frac{\pi}{4}$ and the whole representation of $\phi(n)$ is consequently obtained as

$$\phi(n) = \sqrt{\frac{1}{N}}\big(\cos(kn) + \sin(kn)\big). \tag{B.12}$$

The solutions are equivalent to the base functions of the Hartley transformation.

B.2 Some Deformations of Toda Lattice

B.2.1 Toda's discovery

Here, we review the procedure of the discovery of the Toda potential [89]. The periodic chain with nearest-neighbor interaction can be represented as

$$H = \frac{1}{2m}\sum_{n=1}^{N} p_n^2 + \sum_{n=1}^{N} V(r_n), \tag{B.13}$$

where r_n is difference of nearest-neighbor pair, $q_{i+1} - q_i$. The equation of motion is derived from the Hamiltonian as

$$m\frac{d^2 q_n}{dt^2} = \frac{\partial V(r_n)}{\partial q_n} - \frac{\partial V(r_{n-1})}{\partial q_n}, \tag{B.14}$$

On the other hand, there is another representation of the equation regarding r_i as the canonical coordinate. Then, with the relation between r_i and q_i,

$$q_n = r_0 + r_1 + \cdots + r_{n-1}, \tag{B.15}$$

the kinetic energy is obtained as

$$\sum_n \frac{m}{2}\dot{q}_n = \sum_n \frac{m}{2}\left(\dot{r_0} + \dot{r_1} + \cdots + \dot{r}_{n-1}\right)^2. \tag{B.16}$$

Thus, the conjugate momentum denoted as s_i respecting r_i is given by

$$s_n = \frac{\partial K}{\partial \dot{r}_n} = m\sum_{l=n}^{N-1}\sum_{j=0}^{l}\dot{r}_j. \tag{B.17}$$

Consequently, the equation of motion obeyed the new canonical coordinates is

$$\begin{aligned} \frac{dr_n}{dt} &= -\frac{s_{n+1} - 2s_n + s_{n-1}}{m} \\ \frac{ds_n}{dt} &= -\frac{\partial V(r_n)}{\partial r_n}. \end{aligned} \tag{B.18}$$

Supposing that the r.h.s in the second equation of Eq. (B.18) has an inverse function, the equation of motion is deformed to

$$\frac{d\chi(\dot{s}_n)}{dt} = s_{n+1} - 2s_n + s_{n-1}, \tag{B.19}$$

where $\chi(\dot{s_n})$ is the inverse function.

$$r_n = -\frac{1}{m}\chi(\dot{s_n}). \tag{B.20}$$

Eq. (B.19) is equivalent to the original equation of motion Eq. (B.13). In fact, if one suppose a simple linear function as $\chi(\dot{s}_n)$, it derive the equation of motion of harmonic oscillator chain:

$$m\frac{d^2 s_n}{dt^2} = s_{n+1} - 2s_n + s_{n-1}. \tag{B.21}$$

Obviously, the sine of trigonometric function satisfies Eq. (B.19) and it implies the simple harmonic oscillator. Toda come up with another periodic function satisfying Eq. (B.19), which is Jacobi elliptic functions:

$$Z(u) := \int_0^u \mathrm{dn}^2 w\, dw - \frac{E}{K}u, \tag{B.22}$$

$$Z(u+v) + Z(u-v) - 2Z(u) = \frac{d}{du}\log\left[1 + \frac{1}{1/\mathrm{sn}^2 v - 1 + E/K}\frac{dZ(u)}{du}\right], \tag{B.23}$$

where $Z(u)$ is Jacobi's zeta function, and E and K are complete elliptic integral of the first kind and second kind respectively.

$$K = K(q) = \int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{1 - q^2 \sin^2 \theta}} = \int_0^1 \frac{dx}{\sqrt{(1-x^2)(1-q^2x^2)}} \tag{B.24}$$

$$E = E(q) = \int_0^{\frac{\pi}{2}} \sqrt{1 - q^2 \sin^2 \theta} d\theta = \int_0^1 \sqrt{\frac{1-q^2x^2}{1-x^2}} dx, \tag{B.25}$$

where $|q| < 1$ is called modulus. Hereafter, we don't write the modulus explicitly excepting confusable cases. Furthermore, sn and dn are the elliptic sine and the delta amplitude respectively. By using the elliptic functions, Eq. (B.23) is deformed as

$$\frac{d}{dt} m \log\left[1 + \frac{(mK^2\omega^2/\pi^2)^{-1}}{1/\mathrm{sn}^2(kK/\pi) - 1 + E/K} \dot{s}_n(t)\right] = s_{n+1}(t) + s_{n-1}(t) - 2s_n(t), \tag{B.26}$$

where u,v and s_n are defined as

$$\begin{aligned} u &= 2K(\omega t \pm kn) \\ v &= 2Kk \\ s_n(t) &:= 2mK\omega Z(u), \end{aligned} \tag{B.27}$$

where ω is the frequency, k is the wave number defined by $k_j = 2\pi j/N$, t is time, and b is the constant. As the result, χ is obtained as

$$\chi(\dot{s}_n) = m \log\left[1 + \frac{(mK^2\omega^2/\pi^2)^{-1}}{1/\mathrm{sn}^2(kK/\pi) - 1 + E/K} \dot{s}_n(t)\right] - m\sigma. \tag{B.28}$$

where σ is a constant. According with Eq. (B.20), $\chi(\dot{s}_n)$ must depend on only $\dot{s}_n$. Thus, the prefactor of $\dot{s}_n$ is just a constant, denoted as a. This condition indicates that a nonlinear dispersion relation. To obtain the concrete form of the potential function,

$$\begin{aligned} r_n &= -\log\left(1 + \frac{\dot{s}_n}{a}\right) + \sigma \\ \dot{s}_n &= ae^{-(r_n - \sigma) - 1} = -\frac{\partial V(r_n)}{\partial r_b}, \end{aligned} \tag{B.29}$$

Consequently, the potential function is obtained as

$$V(r) = ae^{-(r-\sigma)} + ar + \mathrm{const}, \tag{B.30}$$

Hereafter, the constants are fix, $a = 1, \sigma = 0, m = 1$.

B.2.2 Lax pair of Toda lattice

The Lax pair of the Toda lattice is given by [92].

$$L := \begin{pmatrix} b_1 & a_1 & & & & & & & a_N \\ a_1 & b_2 & & & & & & & \\ & & \ddots & & & & 0 & & \\ & & & b_{i-1} & a_{i-1} & & & & \\ & & & a_{i-1} & b_i & a_i & & & \\ & & & & a_i & b_{i+1} & & & \\ & & 0 & & & & \ddots & & \\ & & & & & & & b_{N-1} & a_{N-1} \\ a_N & & & & & & & a_{N-1} & b_N \end{pmatrix} \tag{B.31}$$

$$M := \begin{pmatrix} 0 & -a_1 & & & & & & & a_N \\ a_1 & 0 & & & & & & & \\ & & \ddots & & & & 0 & & \\ & & & 0 & -a_{i-1} & & & & \\ & & & a_{i-1} & 0 & -a_i & & & \\ & & & & a_i & 0 & & & \\ & & 0 & & & & \ddots & & \\ & & & & & & & 0 & -a_{N-1} \\ -a_N & & & & & & & a_{N-1} & b_N \end{pmatrix}, \tag{B.32}$$

where a_n and b_n are Flashka variables defined as [97]

$$\begin{aligned} a_i &:= \frac{1}{2} e^{-(q_{i+1}-q_i)}, \\ b_i &:= \frac{1}{2} p_i. \end{aligned} \tag{B.33}$$

B.2.3 Efficient calculation of the action variables

The action-angle variables of the Toda lattice is obtained though the Lax pair [89]. If the discriminants is calculated by the naive method, which obtains it from the determinant directly. The naive strategy is too heavy to iterate the manipulation. In general, the determinant of the matrix is calculated by the LU factorization. The determinant of the product of the matrix is product of the each determinant. The determinant of the triangular matrix is easily calculated by the product of the diagonal elements. The LU decomposition is performed in $O(N^3)$ complexity. Thus, whole process is carried out in $O(N^3)$ complexity. Since the Lax matrix of the Toda lattice has specific property, however, there is more efficient method. The method use the Hill's equation [95, 96]. Considering the diagonalization of the Lax matrix, we obtain the following relation:

$$a_{n-1}\phi(n-1) + b_n\phi(n) + a_n\phi(n+1) = \lambda\phi(n), \ a_{n+N} = a_n, \ b_{n+N} = b_n. \tag{B.34}$$

The equation is regarded as the discretization of the second-order differential equation with periodic external field. Since the equation is similar to Eq. (B.4), $\phi(n)$ is consist of two linearly independent solutions. Thus, we represent the solution as

$$\phi(n) = C_1\phi_1(n) + C_2\phi_2(n), \tag{B.35}$$

where ϕ_1 and ϕ_2 are the independent solutions. For precisness, we use the following initial condition.

$$\phi_1(0) = 1, \ \phi_2(0) = 0, \ \phi_1(1) = 0, \phi_2(1) = 1. \tag{B.36}$$

Through the initial condition, the independent solutions are obtained as

$$\begin{aligned}\phi_1(n) &= -a_0\left(\prod_{j=1}^{n-1} a_j\right)^{-1}\left[\lambda^{n-2} - \left(\sum_{j=2}^{n-1} b_j\right)\lambda^{n-3} + \cdots\right],\\ \phi_2(n) &= \left(\prod_{j=1}^{n-1} a_j\right)^{-1}\left[\lambda^{n-1} - \left(\sum_{j=1}^{n-1} b_j\right)\lambda^{n-2} + \cdots\right].\end{aligned} \tag{B.37}$$

The coefficients of the equations are related to the determinant of the Lax matrix. To see the relation, we introduce the function of eigenvalues defined as

$$\begin{aligned}\Delta(\lambda) &= \phi_1(N) + \phi_2(N+1),\\ &= \left(\prod_{j=1}^{N} a_j\right)^{-1}\left[\lambda^{N} - \left(\sum_{j=1}^{N} b_j\right)\lambda^{N-1} + \cdots\right].\end{aligned} \tag{B.38}$$

It is easily show that the coefficient of λ^0 in the equation is 1. Comparing the determinance of the Lax matrix, the relation between $\Delta(\lambda)$ and determinance is given by

$$\det(L - \lambda I) = (-1)^N\left(\prod_{j=1}^{N} a_j\right)(\Delta(\lambda) - 2)\,. \tag{B.39}$$

Thus, $\Delta(\lambda)$ is nothing but the function called discriminant, which is given in Eq. (2.44).

Appendix C

Sampling Boltzmann Distribution by Hamiltonian Monte Carlo

In this appendix, we review the method to sample the Boltzmann distribution of the Toda lattice. Furthermore, some results of thermodynamics of the Toda lattice are introduced. The calculations are performed for checking the implementation of the Hamiltonian Monte Carlo.

C.1 Algorithm

The Hamiltonian Monte Carlo, or Hybrid Monte Carlo, is one of the methods to sample the distribution of the system described by continuous variables based on the Metropolis-Hastings algorithm [141, 142]. We suppose that the purpose is to obtain the canonical distribution of a potential function V. For convenience, V obeys only a variable q and the dimension is just one. It means that the state of the system is specified the variable q. It is very simple but without loss of generality. If the potential function $V(q)$ is given, the canonical distribution is

$$\rho(q) = \frac{e^{-\beta V(q)}}{Z_V}, \tag{C.1}$$

where β is the inverse temperature and Z_V is the partition function, $Z_V = \mathrm{Tr}\exp(-\beta V(q))$. There are various methods to sample the distribution [157, 158]. One of the versatile algorithm is Metropolis-Hastings algorithm, which is one of the Markov chain Monte Carlo algorithms [159, 160]. In the method, one produce the target distribution as the stationary distribution of the stochastic process. Let q_0 be the initial state of the system. The Metropolis-Hastings algorithm has two steps. In the first step, one produce the candidate of the new state q_1 from the proposal distribution of q_0, denoted as $Q(q_1|q_0)$. Secondly, the candidate is probabilistically accepted as the new state with the acceptance ratio given by

$$a(q_1|q_0) = \min\left(1, \frac{Q(q_0|q_1)\rho(q_1)}{Q(q_1|q_0)\rho(q_0)}\right). \tag{C.2}$$

The ratio is derived from the detailed balance condition which implies that $\rho(q)$ is stationery in the update process. Leaving the problem of the correlation between the states aside, the iteration generates the states which is obeys the distribution $\rho(q)$ and the purpose is achieved consequently.

The efficiency of the algorithm depends on the proposal distribution. If the proposal distribution suggests the candidate far from the distribution, the number of rejection might increase. On the contrary, if the distribution generates the states near the initial one, the correlation becomes large.

In the Hamiltonian Monte Carlo, the proposal distribution is comprised of the Hamiltonian dynamics and Gaussian distribution. The essential point of Hamiltonian Monte Carlo is introduction of the auxiliary variables playing a rule of the “momentum”. Through the “momentum” which is the conjugate variable of q, the algorithm can generate the candidates efficiently. Let p be the conjugate momentum of the variable q. Then, the whole Hamiltonian and the canonical distribution is given by

$$\rho(q,p) = \frac{e^{-\beta H(q,p)}}{Z_H}, \ H := \frac{p^2}{2m} + V(q). \tag{C.3}$$

One can easily derive the original target distribution by tracing out the momentum. For convenience, the pair of variables (q,p) is denoted as x. Through this representation, the Hamiltonian dynamics can be introduced (see Eq. (2.3)). The candidate states are sampled by the following way: (i) sampling the momentum from the Maxwell distribution (ii) evolve the state by the Hamiltonian dynamics. As mentioned below, each step satisfies the detailed balance. For the step (i), the momentum is sampled from the true distribution. The process is obviously rejection-free and satisfies the detailed balance. It can be clarified by calculating the product of the proposal distribution and the target distribution:

$$\begin{aligned} Q(x_1|x_0)\rho(x_0) &= \mathcal{N}(p_1;0,1/\beta)\delta(q_1-q_0)\mathcal{N}(p_0;0,1/\beta)\frac{e^{-\beta V(q_0)}}{Z_V}, \\ &= \mathcal{N}(p_0;0,1/\beta)\delta(q_0-q_1)\mathcal{N}(p_1;0,1/\beta)\frac{e^{-\beta V(q_1)}}{Z_V}, \\ &= Q(x_0|x_1)\rho(x_1). \end{aligned} \tag{C.4}$$

where $\mathcal{N}(p;0,1/\beta)$ is a Gaussian distribution having mean zero and variance $1/\beta$. For the step (ii), the proposal distribution is defined by

$$Q(x_1|x_0) = \delta(x_1 - \phi^t(x_0)), \tag{C.5}$$

where ϕ^t is the exponential map defined by the the Hamiltonian vector filed. Thus, the product $Q(x_1|x_0)\rho(x_0)$ is

$$\begin{aligned} Q(x_1|x_0)\rho(x_0) &= \delta(x_1-\phi^t(x_0))\rho(x_0), \\ &= \frac{1}{|\det J_{\phi^{-t}}|}\delta(x_0-\phi^{-t}(x_1))\rho(x_0), \\ &= \delta(x_0-\phi^t(x_1))\rho(x_1)\frac{\rho(x_0)}{\rho(x_1)}, \\ &= Q(x_0|x_1)\rho(x_1)\frac{\rho(x_0)}{\rho(x_1)}. \end{aligned} \tag{C.6}$$

It is noticed that the determinant of the Jacobian of ϕ^t is one. In the deformation from the second equation to the third equation, we use the time-reversal symmetry of ϕ^t and the Maxwell distribution. In other words, ϕ^{-t} is realized by the time development with flipping the sign of the initial momentum and the Maxwell distribution is invariant with the transformation $p_1 \to -p_1$. Thus, the acceptance ratio is given by

$$a(x_1|x_0) = \min\left(1, \frac{\exp(-\beta H(x_1))}{\exp(-\beta H(x_0))}\right). \tag{C.7}$$

Since the momentum drive the state far from the initial state along the energy surface, the states are likely uncorrelated and acceptable. One can notice that the energy is invariant with the time development and the acceptance ratio always one. In the numerical simulation, however, the time development must not be realized rigorously and the energy does not consist with the initial value. Thus, the time development in the method is implemented by symplectic integrators (see A.1). The numerical integration method is rigorously symplectic and makes the dterminance of the Jacobian be one.

C.2 Boltzmann Distribution of Toda Lattice

In several case, the canonical distribution of the Toda lattice is analytically calculated [89]. In this section, we introduce the two cases with the fixed boundary condition , one particles systems and many particles systems. Hereafter, we suppose the mass is uniform and the strength of the Toda potential is denoted as J.

C.2.1 Two particles case

In the first case, the Hamiltonian is given by

$$H = \frac{p^2}{2} + Je^{-q} + Je^{q}. \tag{C.8}$$

It is also regarded as two particles case with periodic boundary condition ignoring the center of motion. Thus, the canonical distribution is

$$\rho(q,p) = \frac{1}{Z}\exp\left(-\beta\left\{\frac{p^2}{2} + Je^{-q} + Je^{q}\right\}\right), \tag{C.9}$$

. where Z is partition function. In accordance with the basic knowledge of the statistical physics, the thermodynamic properties are known from by the partition function. The partition function is obtained as

$$\begin{aligned} Z &= \mathrm{Tr}\left[\exp\left(-\beta\left\{\frac{p^2}{2} + Je^{-q} + Je^{q}\right\}\right)\right], \\ &= \sqrt{2\beta}K_0(2\beta J). \end{aligned} \tag{C.10}$$

$K_n(x)$ in the r.h.s. of the second equation is the modified Bessel function of the second kind with $n = 0$. Thus, the internal energy of the system is derived as

$$\begin{aligned} U &= -\frac{\partial}{\partial\beta}\log Z, \\ &= \frac{1}{2\beta} + 2J\frac{K_1(2\beta J)}{K_0(2\beta J)}. \end{aligned} \tag{C.11}$$

The numerical calculation by using the Hamiltonian Monte Carlo is showed below.

C.2.2 Many particles case

For convenience, we consider the $N-1$ particles system with the fixed boundary condition. Here, we introduce the boundary coordinate as q_0 and q_N, which means the boundaries fixed at the positions denoted as q_0, q_N. As the partition function of the kinetic term is trivial, we focus on just the partition function obtained from the potential term. The partition function is represented as

$$Z = \int_{-\infty}^{\infty}\prod_{i=1}^{N-1} dq_i \exp\left(-\beta\left\{\sum_{i=0}^{N-1} V(q_{i+1} - q_i)\right\}\right), \tag{C.12}$$

where $V(x)$ is the potential function. If q_0 and q_N are regarded as the variables, Z is a function of them, denoted as $Z(q_0, q_N)$. Instead of calculating the integral directly, we multiply the partition function by a factor of q_0, q_N and integrate over $\{q_i\}_{i=1}^{N}$ [89, 161].

$$\mathcal{Z} = \int_{-\infty}^{\infty}\prod_{i=1}^{N} dq_i Z \exp\left(-\beta p(q_N - q_0)\right), \tag{C.13}$$

where p is a constant. We introduce the other variables $\{r_i\}_{i=1}^{N+1}$ defined by

$$r_i = q_i - q_{i-1},\ i = 1, \cdots, N. \tag{C.14}$$

The Jacobian of the transformation form $\{q_i\}_{i=1}^{N+1}$ to $\{r_i\}_{i=1}^{N}$ is one. Through the variables, one can simplify the integral:

$$\begin{aligned}\mathcal{Z} &= \int_{-\infty}^{\infty} \prod_{i=1}^{N} dr_i \exp\left(-\beta \left\{\sum_{i=1}^{N} V(r_i) + p(r_1 + \cdots + r_N)\right\}\right), \\ &= \left[\int_{-\infty}^{\infty} dr e^{-\beta(V(r)+pr)}\right]^N.\end{aligned} \tag{C.15}$$

The integrated function is evaluated as

$$\int_{-\infty}^{\infty} dr e^{-\beta(J(e^{-r}-1-r)+pr)} = e^{-\beta J} \int_{\infty}^{\infty} dy y^{\beta(p+J)-1} e^{-\beta J y}, \tag{C.16}$$

$$= e^{-\beta J} \frac{\Gamma(\beta(p+J))}{(J\beta)^{\beta(p+J)}}. \tag{C.17}$$

As the result, $\Phi(p)$ is obtained as

$$\mathcal{Z} = \left[e^{-\beta J} \frac{\Gamma(\beta(p+J))}{(J\beta)^{\beta(p+J)}}\right]^N. \tag{C.18}$$

Thus, the Gibs free energy $G(T, N, p)$ is obtained as

$$G(T, N, p) = -\frac{1}{\beta} \ln \mathcal{Z}. \tag{C.19}$$

Since p plays a rule of the pressure, it can be regarded as the partition function of the system which determined by the temperature T, the pressure p and the number of particles N. We denote such system as NTP system. Contrary, the system which is determined N, T and V instead of p is called NVT system. Here, we deal with the system described as chain. Thus, the length of the chain, denoted as L, plays the rule of the volume. The length L is given by $x_0 - x_N$. According with the Legendre transformation from Gibbs free energy to Helmholtz free energy, the free energy of NVT system is obtained as

$$F(\beta, N, L) = \sup_p \left(-pL + G(T, N, p)\right), \tag{C.20}$$

where V is the volume, F is the Helmholtz free energy, and G is the Gibbs free energy. The pressure is determined by the value realizing the r.h.s of Eq. (C.20).

It means that p is the root of the following equation.

$$\begin{aligned}0 &= \frac{\partial}{\partial p}\left(-pL + G(T, N, p)\right), \\ &= -L - \frac{1}{\beta}\frac{\partial}{\partial p} \ln \mathcal{Z}, \\ &= -L - N\left(\frac{d}{dz} \ln \Gamma(z)\bigg|_{\beta(p+J)} - \ln(J\beta)\right).\end{aligned} \tag{C.21}$$

Hereafter, we denote the root of the equation as p^* This equation is also regarded as the evaluation of the inverse Laplacian transformation of $\mathcal{Z}$ by the stationary method [162]:

$$\begin{aligned}Z &= \frac{1}{2\pi i} \int dp\ e^{\beta p L} \mathcal{Z}, \\ &\approx e^{\beta p^* L}\ \mathcal{Z}|_{p=p^*} \frac{1}{2\pi i} \int_{-i\infty+p^*}^{i\infty+p^*} dp\ e^{\frac{(p-p^*)^2}{2} \frac{\partial^2}{\partial p^2}(\beta p L + \ln \mathcal{Z})\big|_{p=p^*}}.\end{aligned} \tag{C.22}$$

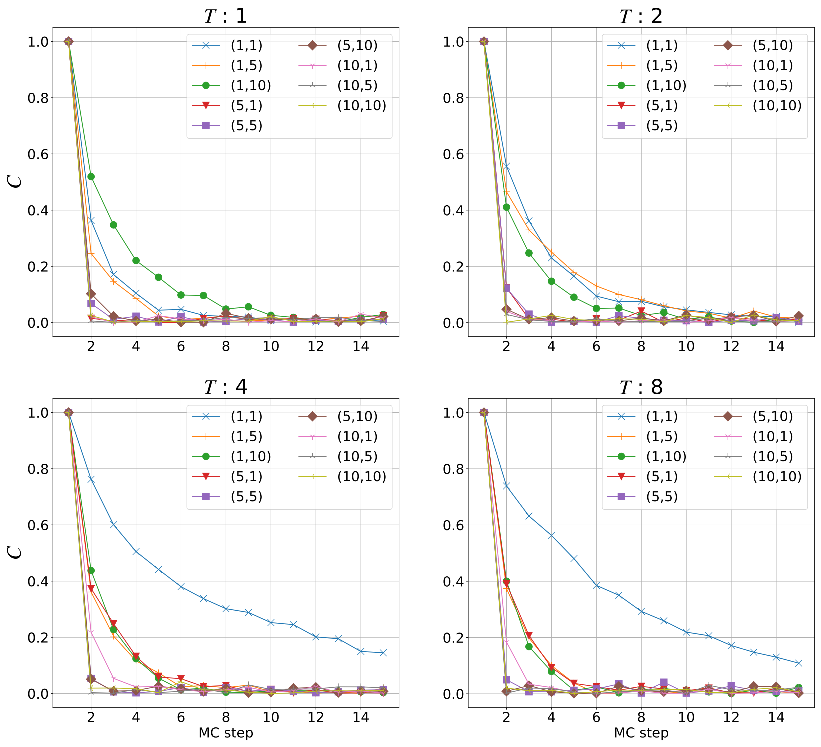


Figure C.1: The autocorrelation function of the potential energies at different relaxation time and temperature. C is the autocorrelation function. The pair numbers in the legend is $(\tau_{\text{cooling}}, \tau_{\text{relax}})$.

According with the correspondence between the partition function and Helmholtz free energy, the internal energy is obtained as

$$\begin{aligned} U &= -\frac{\partial}{\partial \beta} \ln Z, \\ &= \frac{\partial}{\partial \beta}\left(-\beta p^* L - \ln \mathcal{Z}|_{p=p^*}\right), \\ &= -p^* L - N(p^* + J)\left(\frac{d}{dz}\ln \Gamma(z)\bigg|_{\beta(p^*+J)} - \ln(J\beta e)\right) \end{aligned} \tag{C.23}$$

Thus, the internal energy of the periodic Toda lattice is obtained by substituting zero in to L. It is noted that the N particles periodic system has $N-1$ relative coordinates and a boundary. In this sense, the N particles periodic system is $N-1$ fixed boundary system discussed above.

C.2.3 Numerical results

For confirming validity of implementation and finding the appropriate parameters, we compare the numerical results and the analytical calculations. Generally, the correlation between the states sampled from the Markov chain exists and causes the statistical error. There are several approaches to treat the error. Thus, we suppress the correlation by setting the relaxation time and cooling time into sampling. The relaxation time is the time for time development in the HMC. The cooling time is the number of accepts to next sampling. We denote each parameter as τ_{relax} and τ_{cooling} For instance, if the cooling

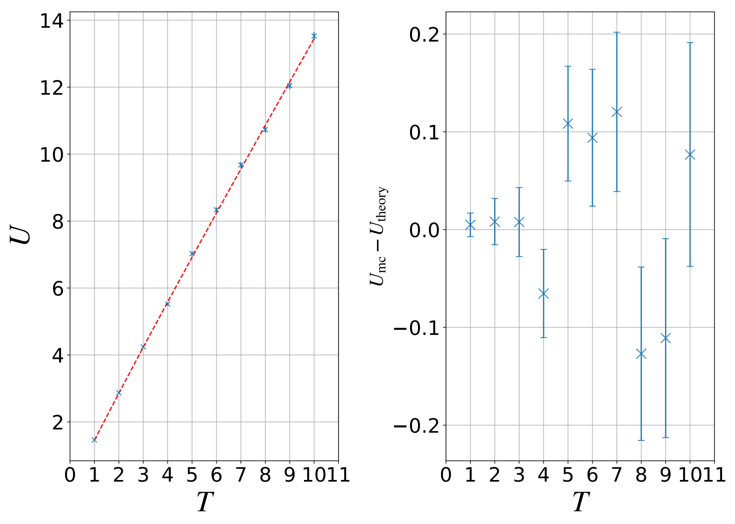


Figure C.2: Internal energy at various temperature. The red line is the analytical calculation with Eq. (C.11). U_{theory} and U_{mc} in the left figure are the internal energy calculated analytically and numerically.

time is 5, we sample the state after 5 accepts. To determine the sufficient relaxation time and cooling time, we observe the autocorrelation function, especially, the autocorrelation function of the potential energy. Furthermore, we measure the internal energy in various temperatures and check the statistical error. Hereafter, we set the interval time of time development $dt = 0.05$ and the initial relaxation time is 10.

First of all, we compare the one particle case. Fig. C.1 shows the autocorrelation function of the potential function with different relaxation time, temperature, and cooling time. If the cooling time is too short, the correlation survive in the Markov chain. Form the observation, we determine the sufficient parameters, $\tau_{\mathrm{relax}} = 5$, $\tau_{\mathrm{cooling}} = 10$. The parameters exhibit that the autocorrelation function decrease immediately at various temperatures. Using the parameters, we calculate the internal energy. The results are shown in Fig. C.2. The red dashed line is obtained from the analytical calculations, Eq. (C.11). The error bar is the confidence interval at 1σ. The probability that the true value in the interval is approximately 68%. In this observation, the values obtained analytically corresponds to it. We can check the behavior by changing the seed. Fig. C.3 shows the values obtained from the numerical calculation by a hundred different seeds. The seeds are obtained from the random bit generator. Within the seeds, 72 out of 100 numerical results include the analytical results in the error bar. The ratio roughly shows the statistical error obeys the theoretical confidence interval. Thus, the parameters and our implementation is valid within the observations.

Next, we focus on the five particles case, which is mainly discussed in the body of our thesis. Fig. C.4 shows also the autocorrelation function of the potential function with different relaxation time, temperature, and cooling time. Within the observation, the parameters, $\tau_{\mathrm{relax}} = 5$, $\tau_{\mathrm{cooling}} = 10$, are sufficient for the five patricles case as well. Using the parameters, we calculate the internal energy. The results are shown in Fig. C.5. The red dashed line is obtained from the analytical calculations, Eq. (C.23). Since we use the stationery method for approximation, the analytical results are not rigorously equal to the true values. The approximation is sufficiently accurate to compare the numerical results. Fig. C.6 shows the values obtained from the numerical calculation by a hundred different seeds. 67 out of 100 numerical results include the analytical results in the error bar within the samples.

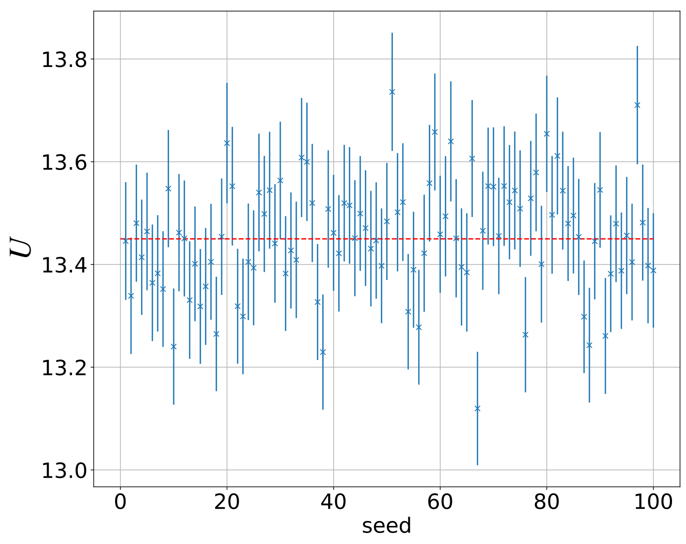


Figure C.3: The numerical results obtained at different seed at temperature $T = 10.0$. The red dashed line shows the value of analytical calculation. The error bar is the confidence interval at 1σ. The horizontal axis does not mean the value of seed, but just the tag number. 72 ouf of 100 results includes the value obtained analytically.

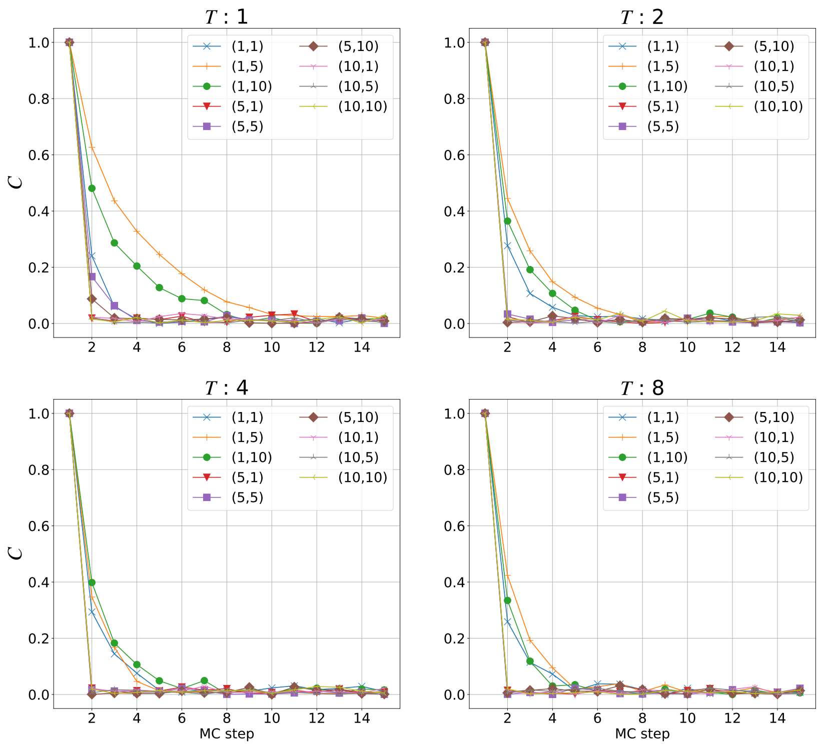


Figure C.4: The autocorrelation function of the potential energies at different relaxation time and temperature. C is the autocorrelation function. The pair numbers in the legend is $(\tau_{\text{cooling}}, \tau_{\text{relax}})$.

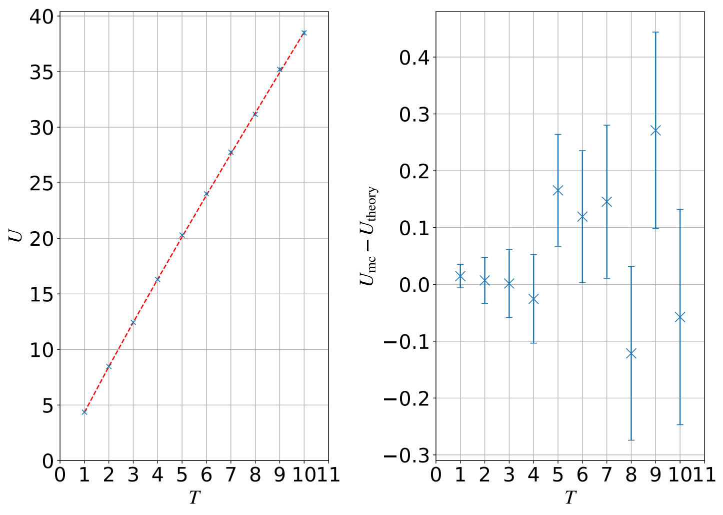


Figure C.5: Internal energy at various temperature. The red dashed line is the analytical calculation with Eq. (C.11). U_{theory} and U_{mc} in the left figure are the internal energy calculated analytically and numerically.

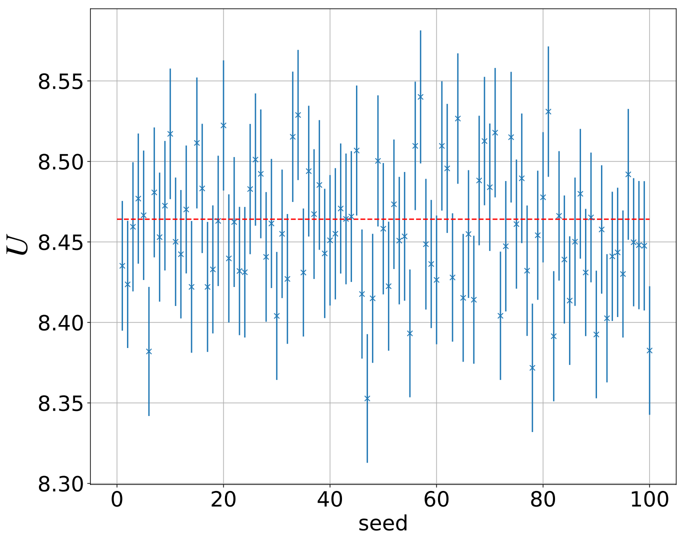


Figure C.6: The numerical results obtained at different seed at temperature $T = 2.0$. The red line shows the value of analytical calculation. The horizontal axis does not mean the value of seed, but just the tag number. The error bar is the confidence interval at 1σ. 72 ouf of 100 results includes the value obtained analytically.

Bibliography

[1] Z. Zhao, P. Zheng, S. Xu, and X. Wu, “Object detection with deep learning: a review”, IEEE Transactions on Neural Networks and Learning Systems **30**, 3212 (2019).

[2] R. Nian, J. Liu, and B. Huang, “A review on reinforcement learning: introduction and applications in industrial process control”, Computers & Chemical Engineering **139**, 106886 (2020).

[3] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent trends in deep learning based natural language processing [review article]”, IEEE Computational Intelligence Magazine **13**, 55 (2018).

[4] H. GM, M. K. Gourisaria, M. Pandey, and S. S. Rautaray, “A comprehensive survey and analysis of generative models in machine learning”, Computer Science Review **38**, 100285 (2020).

[5] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.”, Psychological review **65**, 386 (1958).

[6] G. Cybenko, “Approximation by superpositions of a sigmoidal function”, Mathematics of Control, Signals and Systems **2**, 303 (1989).

[7] K. Hornik, “Approximation capabilities of multilayer feedforward networks”, Neural Networks **4**, 251 (1991).

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (The MIT Press, 2016).

[9] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, “The expressive power of neural networks: a view from the width”, in Advances in neural information processing systems, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (2017), pp. 6231–6239.

[10] C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, and T. B. Schardl, “There’s plenty of room at the top: what will drive computer performance after moore’s law?”, Science **368** (2020).

[11] R. Raina, A. Madhavan, and A. Y. Ng, “Large-scale deep unsupervised learning using graphics processors”, in Proceedings of the 26th annual international conference on machine learning, ICML ’09 (2009), pp. 873–880.

[12] D. Steinkrau, P. Y. Simard, and I. Buck, “Using GPUs for machine learning algorithms”, in Proceedings of the eighth international conference on document analysis and recognition, ICDAR ’05 (2005), pp. 1115–1119.

[13] J. Schmidhuber, “Deep learning in neural networks: an overview”, Neural Networks **61**, 85 (2015).

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors”, Nature **323**, 533 (1986).

[15] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, “Machine learning and the physical sciences”, Rev. Mod. Phys. **91**, 045002 (2019).

[16] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim, “Machine learning in materials informatics: recent applications and prospects”, npj Computational Materials **3**, 54 (2017).

[17] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks", Science **355**, 602 (2017).

[18] D.-L. Deng, X. Li, and S. Das Sarma, "Quantum entanglement in neural network states", Phys. Rev. X **7**, 021021 (2017).

[19] G. Carleo, Y. Nomura, and M. Imada, "Constructing exact representations of quantum many-body systems with deep neural networks", Nature Communications **9**, 5322 (2018).

[20] X. Gao and L.-M. Duan, "Efficient representation of quantum many-body states with deep neural networks", Nature Communications **8**, 662 (2017).

[21] I. A. Balyakin, S. V. Rempel, R. E. Ryltsev, and A. A. Rempel, "Deep machine learning interatomic potential for liquid silica", Phys. Rev. E **102**, 052125 (2020).

[22] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, "Finding density functionals with machine learning", Phys. Rev. Lett. **108**, 253002 (2012).

[23] R. Nagai, R. Akashi, S. Sasaki, and S. Tsuneyuki, "Neural-network Kohn-Sham exchange-correlation potential and its out-of-training transferability", The Journal of Chemical Physics **148**, 241737 (2018).

[24] H. Suwa, J. S. Smith, N. Lubbers, C. D. Batista, G.-W. Chern, and K. Barros, "Machine learning for molecular dynamics with strongly correlated electrons", Phys. Rev. B **99**, 161107 (2019).

[25] Q. Ni, M. Tang, Y. Liu, and Y.-C. Lai, "Machine learning dynamical phase transitions in complex networks", Phys. Rev. E **100**, 052312 (2019).

[26] A. Lidiak and Z. Gong, "Unsupervised machine learning of quantum phase transitions using diffusion maps", Phys. Rev. Lett. **125**, 225701 (2020).

[27] Y. Tomita, K. Shiina, Y. Okabe, and H. K. Lee, "Machine-learning study using improved correlation configuration and application to quantum Monte Carlo simulation", Phys. Rev. E **102**, 021302 (2020).

[28] J. Carrasquilla and R. G. Melko, "Machine learning phases of matter", Nature Physics **13**, 431 (2017).

[29] K. Kashiwa, Y. Kikuchi, and A. Tomiya, "Phase transition encoded in neural network", Progress of Theoretical and Experimental Physics **2019**, 083A04 (2019).

[30] Y. D. Hezaveh, L. P. Levasseur, and P. J. Marshall, "Fast automated analysis of strong gravitational lenses with convolutional neural networks", Nature **548**, 555 (2017).

[31] V. Stanev, C. Oses, A. G. Kusne, E. Rodriguez, J. Paglione, S. Curtarolo, and I. Takeuchi, "Machine learning modeling of superconducting critical temperature", npj Computational Materials **4**, 29 (2018).

[32] S.-M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression", Science Advances **6** (2020).

[33] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, "Discovering physical concepts with neural networks", Phys. Rev. Lett. **124**, 010508 (2020).

[34] L. Huang, Y.-f. Yang, and L. Wang, "Recommender engine for continuous-time quantum Monte Carlo methods", Phys. Rev. E **95**, 031301 (2017).

[35] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, "Self-learning Monte Carlo method", Phys. Rev. B **95**, 041101 (2017).

[36] J. Rogal, E. Schneider, and M. E. Tuckerman, "Neural-network-based path collective variables for enhanced sampling of phase transformations", Phys. Rev. Lett. **123**, 245701 (2019).

[37] S. Li, P. M. Dee, E. Khatami, and S. Johnston, "Accelerating lattice quantum Monte Carlo simulations using artificial neural networks: application to the Holstein model", Phys. Rev. B **100**, 020302 (2019).

[38] L. Huang and L. Wang, “Accelerated Monte carlo simulations with restricted Boltzmann machines”, Phys. Rev. B **95**, 035105 (2017).

[39] B. McNaughton, M. V. Milošević, A. Perali, and S. Pilati, “Boosting Monte Carlo simulations of spin glasses using autoregressive neural networks”, Phys. Rev. E **101**, 053312 (2020).

[40] M. S. Albergo, G. Kanwar, and P. E. Shanahan, “Flow-based generative models for Markov chain Monte Carlo in lattice field theory”, Phys. Rev. D **100**, 034515 (2019).

[41] G. Kanwar, M. S. Albergo, D. Boyda, K. Cranmer, D. C. Hackett, S. Racanière, D. J. Rezende, and P. E. Shanahan, “Equivariant flow-based sampling for lattice gauge theory”, Phys. Rev. Lett. **125**, 121601 (2020).

[42] S.-H. Li and L. Wang, “Neural network renormalization group”, Phys. Rev. Lett. **121**, 260601 (2018).

[43] N. Yoshioka, Y. Akagi, and H. Katsura, “Transforming generalized Ising models into Boltzmann machines”, Phys. Rev. E **99**, 032113 (2019).

[44] F. Noé, S. Olsson, J. Köhler, and H. Wu, “Boltzmann generators: sampling equilibrium states of many-body systems with deep learning”, Science **365**, eaaw1147 (2019).

[45] I. A. Luchnikov, S. V. Vintskevich, D. A. Grigoriev, and S. N. Filippov, “Machine learning non-Markovian quantum dynamics”, Phys. Rev. Lett. **124**, 140502 (2020).

[46] D.-K. Kim, Y. Bae, S. Lee, and H. Jeong, “Learning entropy production via neural networks”, Phys. Rev. Lett. **125**, 140604 (2020).

[47] N. Yoshioka and R. Hamazaki, “Constructing neural stationary states for open quantum many-body systems”, Phys. Rev. B **99**, 214306 (2019).

[48] A. A. Melnikov, P. Sekatski, and N. Sangouard, “Setting up experimental Bell tests with reinforcement learning”, Phys. Rev. Lett. **125**, 160401 (2020).

[49] A. Sheverdin, F. Monticone, and C. Valagiannopoulos, “Photonic inverse design with neural networks: the case of invisibility in the visible”, Phys. Rev. Applied **14**, 024054 (2020).

[50] R. Fournier, L. Wang, O. V. Yazyev, and Q. Wu, “Artificial neural network approach to the analytic continuation problem”, Phys. Rev. Lett. **124**, 056401 (2020).

[51] R. T. D’Agnolo and A. Wulzer, “Learning new physics from a machine”, Phys. Rev. D **99**, 015014 (2019).

[52] G. Arutyunov, *Elements of classical and quantum integrable systems*, 1st ed., 2198-7882 (Springer International Publishing, 2019).

[53] R. A. Barankov, L. S. Levitov, and B. Z. Spivak, “Collective Rabi oscillations and solitons in a time-dependent BCS pairing problem”, Phys. Rev. Lett. **93**, 160401 (2004).

[54] E. A. Yuzbashyan, V. B. Kuznetsov, and B. L. Altshuler, “Integrable dynamics of coupled Fermi-Bose condensates”, Phys. Rev. B **72**, 144524 (2005).

[55] T. Langen, S. Erne, R. Geiger, B. Rauer, T. Schweigler, M. Kuhnert, W. Rohringer, I. E. Mazets, T. Gasenzer, and J. Schmiedmayer, “Experimental observation of a generalized Gibbs ensemble”, Science **348**, 207 (2015).

[56] M. Adler, P. van Moerbeke, and P. Vanhaecke, *Algebraic integrability, Painlevé geometry and Lie algebras*, Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge / A Series of Modern Surveys in Mathematics (Springer Berlin Heidelberg, 2004).

[57] T. Mori, T. N. Ikeda, E. Kaminishi, and M. Ueda, “Thermalization and prethermalization in isolated quantum systems: a theoretical overview”, Journal of Physics B: Atomic, Molecular and Optical Physics **51**, 112001 (2018).

[58] M. Rigol, V. Dunjko, V. Yurovsky, and M. Olshanii, "Relaxation in a completely integrable many-body quantum system: an ab initio study of the dynamics of the highly excited states of 1d lattice hard-core bosons", Phys. Rev. Lett. **98**, 050405 (2007).

[59] L. Vidmar and M. Rigol, "Generalized Gibbs ensemble in integrable lattice models", Journal of Statistical Mechanics: Theory and Experiment **2016**, 064007 (2016).

[60] T. Langen, T. Gasenzer, and J. Schmiedmayer, "Prethermalization and universal dynamics in near-integrable quantum systems", Journal of Statistical Mechanics: Theory and Experiment **2016**, 064009 (2016).

[61] G. P. Berman and F. M. Izrailev, "The Fermi–Pasta–Ulam problem: fifty years of progress", Chaos: An Interdisciplinary Journal of Nonlinear Science **15**, 015104 (2005).

[62] G. Gallavotti, *The Fermi-Pasta-Ulam problem: a status report*, Vol. 728 (Springer, 2007).

[63] G. Benettin and A. Ponno, "Time-scales to equipartition in the Fermi–Pasta–Ulam problem: finite-size effects and thermodynamic limit", Journal of Statistical Physics **144**, 793 (2011).

[64] G. Benettin, H. Christodoulidi, and A. Ponno, "The Fermi-Pasta-Ulam problem and its underlying integrable dynamics", Journal of Statistical Physics **152**, 195 (2013).

[65] W. Fu, Y. Zhang, and H. Zhao, "Universal law of thermalization for one-dimensional perturbed Toda lattices", **21**, 043009 (2019).

[66] A. Ponno, H. Christodoulidi, C. Skokos, and S. Flach, "The two-stage dynamics in the Fermi-Pasta-Ulam problem: from regular to diffusive behavior", Chaos: An Interdisciplinary Journal of Nonlinear Science **21**, 043127 (2011).

[67] H. Spohn, "Generalized Gibbs ensembles of the classical Toda chain", Journal of Statistical Physics **180**, 4 (2020).

[68] X. Cao, V. B. Bulchandani, and H. Spohn, "The GGE averaged currents of the classical Toda chain", Journal of Physics A: Mathematical and Theoretical **52**, 495003 (2019).

[69] T. Goldfriend and J. Kurchan, "Equilibration of quasi-integrable systems", Phys. Rev. E **99**, 022146 (2019).

[70] M. Toda, "Vibration of a chain with nonlinear interaction", Journal of the Physical Society of Japan **22**, 431 (1967).

[71] M. Toda, "Wave propagation in anharmonic lattices", Journal of the Physical Society of Japan **23**, 501 (1967).

[72] F. Calogero, "Exactly solvable one-dimensional many-body problems", Lettere al Nuovo Cimento (1971-1985) **13**, 411 (1975).

[73] F. Calogero, "On a functional equation connected with integrable many-body problems", Lettere al Nuovo Cimento (1971-1985) **16**, 77 (1976).

[74] J Moser, "Three integrable Hamiltonian systems connected with isospectral deformations", Advances in Mathematics **16**, 197 (1975).

[75] A. Perelomov, *Integrable systems of classical mechanics and Lie algebras volume I*, Vol. 1 (Birkhäuser Basel, 1990).

[76] M. Cariglia, "Hidden symmetries of dynamics in classical and quantum physics", Rev. Mod. Phys. **86**, 1283 (2014).

[77] V. Arnol'd, *Mathematical methods of classical mechanics*, 2nd, Vol. 60, Graduate Texts in Mathematics (Springer-Verlag New York, 1989).

[78] S.-H. Li, C.-X. Dong, L. Zhang, and L. Wang, "Neural canonical transformation with symplectic flows", Phys. Rev. X **10**, 021020 (2020).

[79] R. Bondesan and A. Lamacraft, “Learning symmetries of classical integrable systems”, ArXiv, 1906.04645 (2019).

[80] I. Kobyzev, S. Prince, and M. Brubaker, “Normalizing flows: an introduction and review of current methods”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1 (2020).

[81] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, in *Advances in neural information processing systems 31*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018), pp. 6571–6583.

[82] J. M. Sanz-Serna, “Symplectic Runge–Kutta schemes for adjoint equations, automatic differentiation, optimal control, and more”, SIAM Review **58**, 3 (2016).

[83] N. S. Morikazu Toda Ryogo Kubo, *Equilibrium statistical mechanics*, 2nd ed., 0171-1873 (Springer-Verlag Berlin Heidelberg, 1992).

[84] Arvind, B. Dutta, N. Mukunda, and R. Simon, “The real symplectic groups in quantum mechanics and optics”, Pramana **45**, 471 (1995).

[85] K. Iwasawa, “On some types of topological groups”, Annals of Mathematics **50**, 507 (1949).

[86] M. de Gosson, *Symplectic geometry and quantum mechanics* (Birkhäuser Basel, 2006).

[87] A. Giorgilli, *Sistemi hamiltoniani e teoria delle perturbazioni - chapter 3: integrable systems*, `http://www.mat.unimi.it/users/antonio/hamsys/Hamsys_3.pdf`.

[88] P. D. Lax, “Integrals of nonlinear equations of evolution and solitary waves”, Communications on Pure and Applied Mathematics **21**, 467 (1968).

[89] M. Toda, *Theory of nonlinear lattices*, 2nd (Springer, Berlin, Heidelberg, 1989).

[90] S. V. Manakov, “Complete integrability and stochastization of discrete dynamical systems”, Soviet Journal of Experimental and Theoretical Physics **40**, 269 (1975).

[91] O. Babelon and C.-M. Viallet, “Hamiltonian structures and Lax equations”, Physics Letters B **237**, 411 (1990).

[92] H. Flaschka, “The Toda lattice. II. existence of integrals”, Phys. Rev. B **9**, 1924 (1974).

[93] M. Olshanetsky and A. Perelomov, “Classical integrable finite-dimensional systems related to Lie algebras”, Physics Reports **71**, 313 (1981).

[94] R. Hirota, “Exact solution of the Korteweg—de Vries equation for multiple collisions of solitons”, Phys. Rev. Lett. **27**, 1192 (1971).

[95] E. Date and S. Tanaka, “Periodic Multi-Soliton Solutions of Korteweg-de Vries Equation and Toda Lattice”, Progress of Theoretical Physics Supplement **59**, 107 (1976).

[96] H. Flaschka and D. W. McLaughlin, “Canonically conjugate variables for the Korteweg-de Vries equation and the Toda lattice with periodic boundary conditions”, Progress of Theoretical Physics **55**, 438 (1976).

[97] H. Flaschka, “On the Toda lattice. II: inverse-scattering solution”, Progress of Theoretical Physics **51**, 703 (1974).

[98] W. Ferguson, H. Flaschka, and D. McLaughlin, “Nonlinear normal modes for the Toda chain”, Journal of Computational Physics **45**, 157 (1982).

[99] A Henrici and T Kappeler, in *Integrable systems and random matrices: in honor of percy deift.* Edited by J Baik, Contemporary Mathematics 458 (American Mathematical Society, Providence, RI, 2008), pp. 11–19.

[100] A. Henrici, “Symmetries of the periodic Toda lattice, with an application to normal forms and perturbations of the lattice with Dirichlet boundary conditions”, Discrete & Continuous Dynamical Systems - A **35**, 2949 (2015).

[101] O. I. Bogoyavlensky, “On perturbations of the periodic Toda lattice”, Comm. Math. Phys. **51**, 201 (1976).

[102] V. Inozemtsev, “On the motion of classical integrable systems of interacting particles in an external field”, Physics Letters A **98**, 316 (1983).

[103] M. Adler, “Some finite dimensional integrable systems and their scattering behavior”, Communications in Mathematical Physics **55**, 195 (1977).

[104] L. P. Eisenhart, “Dynamical trajectories and geodesics”, Annals of Mathematics **30**, 591 (1928).

[105] M. Adler, “On a trace functional for formal pseudo-differential operators and the symplectic structure of the Korteweg-devries type equations”, Inventiones mathematicae **50**, 219 (1978).

[106] B. Kostant, “The solution to a generalized Toda lattice and representation theory”, Advances in Mathematics **34**, 195 (1979).

[107] W. Symes, “Systems of Toda type, inverse spectral problems, and representation theory.”, Inventiones mathematicae **59**, 13 (1980).

[108] C. M. Bishop, *Pattern recognition and machine learning (information science and statistics)* (Springer-Verlag, Berlin, Heidelberg, 2006).

[109] A. Shrestha and A. Mahmood, “Review of deep learning algorithms and architectures”, IEEE Access **7**, 53040 (2019).

[110] S. Elfwing, E. Uchibe, and K. Doya, “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning”, Neural Networks **107**, Special issue on deep reinforcement learning, 3 (2018).

[111] S. Gunasekar, J. D. Lee, D. Soudry, and N. Srebro, “Implicit bias of gradient descent on linear convolutional networks”, in Advances in neural information processing systems, Vol. 31, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (2018), pp. 9461–9471.

[112] S. Gunasekar, B. E. Woodworth, S. Bhojanapalli, B. Neyshabur, and N. Srebro, “Implicit regularization in matrix factorization”, in Advances in neural information processing systems, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (2017), pp. 6151–6159.

[113] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro, “The implicit bias of gradient descent on separable data”, Journal of Machine Learning Research **19**, 1 (2018).

[114] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in 2016 ieee conference on computer vision and pattern recognition (cvpr) (2016), pp. 770–778.

[115] D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization”, ArXiv, 1412.6980 (2017).

[116] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks”, ArXiv, 1406.2661 (2014).

[117] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real NVP”, in 5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings (2017).

[118] D. P. Kingma and P. Dhariwal, “Glow: generative flow with invertible 1x1 convolutions”, in Advances in neural information processing systems (2018), pp. 10215–10224.

[119] T. Teshima, I. Ishikawa, K. Tojo, K. Oono, M. Ikeda, and M. Sugiyama, “Coupling-based invertible neural networks are universal diffeomorphism approximators”, Advances in Neural Information Processing Systems **33** (2020).

[120] S. Kullback and R. A. Leibler, “On information and sufficiency”, Ann. Math. Statist. **22**, 79 (1951).

[121] R. Cabrera, T. Strohecker, and H. Rabitz, “The canonical coset decomposition of unitary matrices through Householder transformations”, Journal of Mathematical Physics **51**, 082101 (2010).

[122] S. Greydanus, M. Dzamba, and J. Yosinski, “Hamiltonian neural networks”, in Advances in neural information processing systems, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (2019), pp. 15379–15389.

[123] A. Choudhary, J. F. Lindner, E. G. Holliday, S. T. Miller, S. Sinha, and W. L. Ditto, “Physics-enhanced neural networks learn order and chaos”, Phys. Rev. E **101**, 062207 (2020).

[124] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, “Lagrangian neural networks”, ArXiv, 2003.04630 (2020).

[125] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou, “Symplectic recurrent neural networks”, ArXiv, 1909.13334 (2020).

[126] Y. Tong, S. Xiong, X. He, G. Pan, and B. Zhu, “Symplectic neural networks in Taylor series form for Hamiltonian systems”, ArXiv, 2005.04986 (2020).

[127] Y. D. Zhong, B. Dey, and A. Chakraborty, “Symplectic ODE-Net: learning Hamiltonian dynamics with control”, ArXiv, 1909.12077 (2020).

[128] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: a review and new perspectives”, IEEE Transactions on Pattern Analysis and Machine Intelligence **35**, 1798 (2013).

[129] N. T. Zung, “Convergence versus integrability in Birkhoff normal form”, Annals of Mathematics **161**, 141 (2005).

[130] H. Ito, “Convergence of Birkhoff normal forms for integrable systems”, Commentarii Mathematici Helvetici **64**, 412 (1989).

[131] T. Kappeler, Y. Kodama, and A. Némethi, “On the Birkhoff normal form of a completely integrable hamiltonian system near a fixed point with resonance”, Annali della Scuola Normale Superiore di Pisa - Classe di Scienze **Ser. 4, 26**, 623 (1998).

[132] H. Yoshida, “Construction of higher order symplectic integrators”, Physics Letters A **150**, 262 (1990).

[133] A. Sehanobish, H. Corzo, O. Kara, and D. V. Dijk, “Learning potentials of quantum systems using deep neural networks”, ArXiv, 2006.13297 (2020).

[134] H. Webber, P. Martre, S. Asseng, B. Kimball, J. White, M. Ottman, G. W. Wall, G. De Sanctis, J. Doltra, R. Grant, B. Kassie, A. Maiorano, J. E. Olesen, D. Ripoche, E. E. Rezaei, M. A. Semenov, P. Stratonovitch, and F. Ewert, “Canopy temperature for simulation of heat stress in irrigated wheat in a semi-arid environment: a multi-model comparison”, Field Crops Research **202**, Modeling crops from genotype to phenotype in a changing climate, 21 (2017).

[135] M. Göçken, M. Özçalıcı, A. Boru, and A. T. Dosdoğru, “Integrating metaheuristics and artificial neural networks for improved stock price prediction”, Expert Systems with Applications **44**, 320 (2016).

[136] M. Despotovic, V. Nedic, D. Despotovic, and S. Cvetanovic, “Evaluation of empirical models for predicting monthly mean horizontal diffuse solar radiation”, Renewable and Sustainable Energy Reviews **56**, 246 (2016).

[137] J. Liu, L. Sun, Q. Li, J. Ming, Y. Liu, and H. Xiong, “Functional zone based hierarchical demand prediction for bike system expansion”, in Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining, KDD ’17 (2017), pp. 957–966.

[138] Y. Zhou and Y. Huang, “Place representation based bike demand prediction”, in 2019 ieee international conference on big data (big data) (2019), pp. 1577–1586.

[139] S. Van Canneyt, P. Leroux, B. Dhoedt, and T. Demeester, "Modeling and predicting the popularity of online news based on temporal and content-related features", Multimedia Tools and Applications **77**, 1409 (2018).

[140] R. Neuneier and H. G. Zimmermann, "How to train neural networks", in *Neural networks: tricks of the trade*, edited by G. B. Orr and K.-R. Müller (Springer Berlin Heidelberg, Berlin, Heidelberg, 1998), pp. 373–423.

[141] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo", Physics Letters B **195**, 216 (1987).

[142] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of Markov chain Monte Carlo* (CRC press, 2011).

[143] L. E. Reichl, *The transition to chaos* (Springer, New York, NY, 1993).

[144] A. Seko, T. Maekawa, K. Tsuda, and I. Tanaka, "Machine learning with systematic density-functional theory calculations: application to melting temperatures of single- and binary-component solids", Phys. Rev. B **89**, 054303 (2014).

[145] T. Ueno, T. D. Rhone, Z. Hou, T. Mizoguchi, and K. Tsuda, "COMBO: an efficient Bayesian optimization library for materials science", Materials Discovery **4**, 18 (2016).

[146] J. Dormand and P. Prince, "A family of embedded Runge-Kutta formulae", Journal of Computational and Applied Mathematics **6**, 19 (1980).

[147] P. Bogacki and L. Shampine, "A 3(2) pair of Runge-Kutta formulas", Applied Mathematics Letters **2**, 321 (1989).

[148] J. R. Cash and A. H. Karp, "A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides", ACM Trans. Math. Softw. **16**, 201 (1990).

[149] J. Sanz-Serna, "Symplectic Runge-Kutta and related methods: recent results", Physica D: Nonlinear Phenomena **60**, 293 (1992).

[150] S. Blanes and P. Moan, "Practical symplectic partitioned Runge–Kutta and Runge–Kutta–Nyström methods", Journal of Computational and Applied Mathematics **142**, 313 (2002).

[151] S. Gan, Z. Shang, and G. Sun, "A class of symplectic partitioned Runge–Kutta methods", Applied Mathematics Letters **26**, 968 (2013).

[152] D. Donnelly and E. Rogers, "Symplectic integrators: an introduction", American Journal of Physics **73**, 938 (2005).

[153] J Candy and W Rozmus, "A symplectic integration algorithm for separable Hamiltonian functions", Journal of Computational Physics **92**, 230 (1991).

[154] E. Forest and R. D. Ruth, "Fourth-order symplectic integration", Physica D: Nonlinear Phenomena **43**, 105 (1990).

[155] F. Neri, *Lie algebras and canonical integration* (Dept. of Physics, University of Maryland, preprint, 1987).

[156] N. Biggs and P. Biggs, *Discrete mathematics*, Oxford science publications (OUP Oxford, 2002).

[157] D. Landau and K. Binder, *A guide to Monte Carlo simulations in statistical physics*, Fourth (Cambridge University Press, USA, 2015).

[158] D. Kroese, T. Taimre, and Z. Botev, *Handbook of Monte Carlo methods*, Wiley Series in Probability and Statistics (Wiley, 2011).

[159] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications", Biometrika **57**, 97 (1970).

[160] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines", The Journal of Chemical Physics **21**, 1087 (1953).

[161] H. Takahashi, “A simple method for treating the statistical mechanics of one-dimensional substances: from proceedings of the Physico-Mathematical Society of Japan (Nippon Suugaku-Buturigakkwai Kizi Tokyo) 24, 60 (1942)”, edited by E. H. Lieb and D. C. Mattis, 25 (1966).

[162] V. N. Likhachev, T. Y. Astakhova, W. Ebeling, and G. A. Vinogradov, “Equilibrium thermodynamics and thermodynamic processesin nonlinear systems”, The European Physical Journal B **72**, 247 (2009).