mathematics

MDPI

Article

Quadrature Based Neural Network Learning of Stochastic Hamiltonian Systems

Xupeng Cheng [1], Lijin Wang [1] and Yanzhao Cao [2,*]

[1] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China; chengxupeng21@mails.ucas.ac.cn (X.C.); ljwang@ucas.ac.cn (L.W.)
[2] Department of Mathematics and Statistics, Auburn University, Auburn, AL 36849, USA
* Correspondence: yzc0009@auburn.edu

Abstract: Hamiltonian Neural Networks (HNNs) provide structure-preserving learning of Hamiltonian systems. In this paper, we extend HNNs to structure-preserving inversion of stochastic Hamiltonian systems (SHSs) from observational data. We propose the quadrature-based models according to the integral form of the SHSs' solutions, where we denoise the loss-by-moment calculations of the solutions. The integral pattern of the models transforms the source of the essential learning error from the discrepancy between the modified Hamiltonian and the true Hamiltonian in the classical HNN models into that between the integrals and their quadrature approximations. This transforms the challenging task of deriving the relation between the modified and the true Hamiltonians from the (stochastic) Hamilton–Jacobi PDEs, into the one that only requires invoking results from the numerical quadrature theory. Meanwhile, denoising via moments calculations gives a simpler data fitting method than, e.g., via probability density fitting, which may imply better generalization ability in certain circumstances. Numerical experiments validate the proposed learning strategy on several concrete Hamiltonian systems. The experimental results show that both the learned Hamiltonian function and the predicted solution of our quadrature-based model are more accurate than that of the corrected symplectic HNN method on a harmonic oscillator, and the three-point Gaussian quadrature-based model produces higher accuracy in long-time prediction than the Kramers–Moyal method and the numerics-informed likelihood method on the stochastic Kubo oscillator as well as other two stochastic systems with non-polynomial Hamiltonian functions. Moreover, the Hamiltonian learning error $\varepsilon_{\mathcal{H}}$ arising from the Gaussian quadrature-based model is lower than that from Simpson's quadrature-based model. These demonstrate the superiority of our approach in learning accuracy and long-time prediction ability compared to certain existing methods and exhibit its potential to improve learning accuracy via applying precise quadrature formulae.

Keywords: stochastic Hamiltonian systems; Hamiltonian Neural Networks; Gaussian numerical quadrature; Simpson's formula

MSC: 65C30; 65P10; 62M45; 37M10

Citation: Cheng, X.; Wang, L.; Cao, Y. Quadrature Based Neural Network Learning of Stochastic Hamiltonian Systems. *Mathematics* **2024**, *12*, 2438. https://doi.org/10.3390/math12162438

Academic Editors: Guang Lin, Zecheng Zhang and Christian Moya

Received: 30 June 2024
Revised: 31 July 2024
Accepted: 3 August 2024
Published: 6 August 2024

1. Introduction

The study of complex physical systems described by Hamiltonian mechanics often requires a deep understanding of their underlying dynamics. The classical Hamiltonian formulation provides a mathematical framework for characterizing the evolution of systems in a deterministic manner. However, in many realworld scenarios, the dynamics are subject to unpredictable fluctuations and noises, rendering traditional Hamiltonian models insufficient to capture their behavior accurately. To address this challenge, the emerging field of stochastic Hamiltonian systems (SHSs) inference is gaining attention [1–3]. Stochastic Hamiltonians account for the inherent randomness and uncertainties present in physical systems, allowing for a more comprehensive modeling approach. The application of

stochastic Hamiltonian systems is prevalent in the real world. As described in [4], Langevin introduced the Langevin equation, a stochastic Hamiltonian system, to explain the motion of particles in the fluid caused by random pulses of fluid molecules colliding and to show that the viscous drag is the average effect of these pulses. In the realm of control theory, stochastic Hamiltonian systems play an important role in the extension of optimal control to stochastic processes [5]. These systems also find applications in biology. For instance, in the study of ion channel selectivity in biological systems, Seifi et al. [6] employed the Spin–Boson model and a stochastic Hamiltonian model under classical noise to simulate the behavior of ion channels. Inferring these stochastic Hamiltonian systems from observed data has proven to be a complex task due to the nonlinearity and high-dimensional nature of the underlying dynamics. In recent years, advancements in deep learning and neural networks have opened up new avenues for tackling this problem. This paper focuses on the application of neural networks to infer SHSs from available data, aiming to learn the underlying dynamics of the observed processes. The goal is to estimate the drift and diffusion Hamiltonian functions that govern the evolution of the SHSs.

Remarkable advancements have been achieved in the realm of learning deterministic Hamiltonian systems. Chen et al. [7] introduced Neural ODE as a method for learning ordinary differential equations (ODEs) from observed time series of states. These states are assumed to be generated by the underlying ODE flow. The approach utilizes adjoint augmented ODE solvers to compute the gradient of the loss function with respect to the network parameters. For Hamiltonian systems, Greydanus et al. [8] proposed the Hamiltonian Neural Networks (HNNs). HNNs employ a neural network $H_{net}(p, q)$ to learn the Hamiltonian function $H(p, q)$ instead of directly learning the vector field so that the symplectic structure as well as certain conserved quantities related to the structure of the underlying system can be extracted from data. The network assumes access to time derivatives $\dot{p}$ and $\dot{q}$ and is trained to match these derivatives along the phase flow. In their work [9], Chen et al. introduced Symplectic Recurrent Neural Networks (SRNNs), which combine the principles of Neural ODE and recurrent neural networks to learn the Hamiltonian from time series data in the state space. SRNNs exhibit robustness and demonstrate good performance in learning separable Hamiltonian dynamics. Zhu et al. [10] proved that Hamiltonian networks based on symplectic integrators possess network targets and exhibit superior generalization ability and prediction accuracy compared to non-symplectic networks. Jin et al. [11] developed SympNets, a novel class of symplectic neural networks for learning Hamiltonian systems. SympNets can simulate any symplectic mappings based on appropriate activation functions. In their work [12], Xiong et al. addressed the problem of learning nonseparable Hamiltonian systems using neural ODE with an augmented Hamiltonian system. Tong et al. [13] developed a lightweight and efficient symplectic neural network based on Taylor series expansion and a fourth-order symplectic Runge-Kutta integrator. This network is also designed to learn separable Hamiltonian systems. In [14], David et al. developed a symplectic learning approach for HNNs. They apply symplectic schemes in the loss function to train a modified Hamiltonian and then deduce the true Hamiltonian using its relationship with the modified one. Chen and Tao proposed the learning of exactly-symplectic maps in [15].

In recent years, a variety of approaches inferring stochastic differential equations (SDEs) from data have arisen. Based on the framework of [7], several authors extended the method to neural stochastic differential equations (neural SDEs) by incorporating stochasticity into the dynamics. Tzen et al. [16] introduced a novel framework that combines neural networks and Gaussian latent variables to model SDEs in the diffusion limit. Ref. [17] presented a method for efficiently computing gradients and performing variational inference in models involving SDEs. Yildiz et al. [18] combined Gaussian processes and neural networks to estimate the drift and diffusion coefficients in SDEs from sparse and noisy data. Jia et al. [19] introduced neural jump SDEs, which integrate neural networks with jump diffusion processes to model rare events and non-Gaussian dynamics. Kong et al. [20] introduced the SDE-Net, treating the forward propagation of a deep neural network as the

evolution of a stochastic dynamic system governed by the Brownian motion. By adopting the generative model of the SDE framework, they successfully addressed various image-related challenges. Archibald et al. [21] presented a sample-wise back-propagation method for training stochastic neural networks through a stochastic optimal control framework, utilizing a discretized stochastic differential equation structure, and provided convergence analysis with and without convexity assumptions. Gobet et al. [22] developed a neural network-based method to estimate the parameters of scalar SDEs from low-frequency data, improving traditional estimation techniques. Ref. [23] introduced a novel stochastic differential equation framework for generative modeling that smoothly transforms data distributions using noise injection, leveraging score-based generative modeling and diffusion probabilistic modeling to achieve record-breaking performance in image generation tasks. Liu et al. [24] proposed a supervised learning framework for training generative models using a novel score-based diffusion model, avoiding issues of unsupervised training by generating labeled data through a training-free score estimation method, resulting in improved sampling efficiency and performance. Meanwhile, some scholars explored the integration of popular models with SDEs. Ref. [25] treated neural SDEs as a Bayesian neural network with infinite depth, utilizing Bayesian inference to infer the drift and diffusion coefficients. Kidger et al. [26] employed neural SDEs as the generator of a GAN while using neural controlled differential equation (CDE) as the discriminator. These methods parameterize the drift and diffusion coefficients with neural networks, but different perspectives and inference methods have led to the development of various neural SDE models, such as those based on the Fokker–Planck equations associated with the underlying SDEs [27–32] where [28–31] extended the driving process to non-Gaussian Lévy processes. There are also other approaches leveraging the probability density of the latent processes for inference such as the variational approaches [33,34], the numeric-informed maximum likelihood method [35], and the normalizing flow techniques [36–40]. Methods extending the ResNet approach and flow map operators to stochastic context emerge in [41,42], etc., and employing neural ODE via deriving dynamics of moments of SDEs' solutions can be found in [43].

For stochastic Hamiltonian systems, we propose a neural network learning approach for extracting the drift and diffusion Hamiltonian functions from observational data, based on expressing the solution in an integral form, and then applying numerical quadrature formulae to approximate them. To deal with the stochasticity, we employ the moment calculations of the solutions. Our contribution mainly lies in the generalization of the HNNs to stochastic context, with a quadrature-based model structure that can leverage numerical quadrature theory to improve the accuracy of HNN learning. Meanwhile, moments calculations provide a direct and simple denoised loss and training strategy which may benefit algorithm generalization. Numerical experiments on four Hamiltonian systems give support to the proposed learning approach and showcase that it has better learning accuracy and generalization ability compared to certain existing methods, and suggest its potential for improving the learning effect by utilizing quadrature with higher accuracy.

The rest of the contents are organized as follows. Section 2 provides an introduction to the problem statement relevant to the stochastic Hamiltonian systems. Section 3 presents the design of the learning model, and network settings including the data acquisition, learning strategy, as well as the derivation of the loss functions. Section 4 conducts numerical experiments on four different Hamiltonian systems, illustrating the performance of the learning algorithm. Section 5 lists some limitations of the current study, followed by a brief conclusion in Section 6.

2. Problem Statement

Consider stochastic Hamiltonian systems [1,2], which are Stratonovich SDEs of the following form

$$d\boldsymbol{y}_t = J^{-1}\nabla\mathcal{H}_0(\boldsymbol{y}_t)dt + J^{-1}\nabla\mathcal{H}_1(\boldsymbol{y}_t)\circ dW(t),\ \boldsymbol{y}(0) = \boldsymbol{y}_0, \tag{1}$$

where $\boldsymbol{y}_t \in \mathbb{R}^d$ $(d = 2m)$, $\mathcal{H}_r(\boldsymbol{y})$ $(r = 0, 1)$ are smooth functions of $\boldsymbol{y}$ which are unknown, $W(t)$ is a standard real-valued Wiener process defined on a complete filtered probability space $\{\Omega, \{\mathcal{F}_t\}_{t\geq 0}, P\}$ and $J = \begin{pmatrix} 0 & I_m \\ -I_m & 0 \end{pmatrix}$ with I_m being the m-dimensional identity matrix. It is proved that, almost surely, the phase flow of system (1) preserves the symplectic structure [2].

Given observational data of $\boldsymbol{y}_t$ on discrete time points in the time interval $t \in [0, T]$, we aim to detect the drift and diffusion Hamiltonian functions $H_r(\boldsymbol{y})$ $(r = 0, 1)$ via neural networks, where the sign of $\mathcal{H}_1(\boldsymbol{y})$ or $\nabla\mathcal{H}_1(\boldsymbol{y})$ is supposed to be known (see, e.g., [35] for similar prerequisite).

3. Methodology

3.1. Improving HNNs from a Quadrature Point of View

As mentioned above, one approach to improving the Hamiltonian Neural Networks (HNNs) for structure-preserving learning of Hamiltonian systems is to use symplectic integrators in the loss functions, namely to employ the Symplectic Hamiltonian Neural Networks (SHNNs) [9,10,12]. However, there remains a certain discrepancy between the learned and the true Hamiltonian functions. To minimize this error, [14] proposed to correct the learned Hamiltonian $\breve{\mathcal{H}}_{net}$ according to the relationship between its theoretical value $\breve{\mathcal{H}}$ the true Hamiltonian $\mathcal{H}$, which is represented by the following Hamilton–Jacobi PDE [44] when the midpoint rule is applied as the symplectic integrator in the loss function:

$$\frac{\partial \breve{\mathcal{H}}}{\partial t}(\boldsymbol{w}, t) = \mathcal{H}\left(\boldsymbol{w} + \frac{1}{2}J^{-1}\nabla_{\boldsymbol{w}}\breve{\mathcal{H}}(\boldsymbol{w}, t)\right), \tag{2}$$

where $\boldsymbol{w} = \frac{\boldsymbol{y}(0)+\boldsymbol{y}(t)}{2}$. For the PDE (2), Feng et al. [45] proposed a solution in the form of a power series of t which was substituted into (2) to determine the unknown coefficients in the series by comparing like powers of t on both sides of the equation, after Taylor expanding the right-hand side of (2) at $\boldsymbol{w}$. Thus, one can obtain (replacing t by the time step h of the midpoint method)

$$\breve{\mathcal{H}}(\boldsymbol{w}, h) = hG_1(\boldsymbol{w}) + h^3G_3(\boldsymbol{w}) + \cdots + h^{2r-1}G_{2r-1}(\boldsymbol{w}) + \ldots, \tag{3}$$

where $G_1(\boldsymbol{w}) = \mathcal{H}(\boldsymbol{w})$, $G_3(\boldsymbol{w}) = \frac{1}{24}\nabla^2\mathcal{H}(\boldsymbol{w})\big(J^{-1}\nabla\mathcal{H}(\boldsymbol{w}), J^{-1}\nabla\mathcal{H}(\boldsymbol{w})\big)$, and so on. To obtain a higher-order truncated approximation of the series, one needs complicated calculations to find $G_{2r-1}(\boldsymbol{w})$ $(r \geq 3)$. Although symbolic computation programs can give the inverted relation that represents $\mathcal{H}$ by $\breve{\mathcal{H}}$ from the truncated series, as was employed in [14], higher order formulations of $\mathcal{H}$ in terms of $\breve{\mathcal{H}}$ as well as its partial derivatives are fairly difficult to obtain, which hinders obtaining higher-order approximations of $\mathcal{H}$ based on modifying $\breve{\mathcal{H}}$.

On the other hand, when we generalize the corrected SHNNs in [14] to structure-preserving learning of stochastic Hamiltonian systems, the relation between $\breve{\mathcal{H}}$ and $\mathcal{H}_0$, $\mathcal{H}_1$ lies in the stochastic Hamilton–Jacobi PDE [3,46,47]

$$\partial_t\breve{\mathcal{H}}(\boldsymbol{w}, t, \omega) = \mathcal{H}_0\left(\boldsymbol{w} + \frac{1}{2}J^{-1}\nabla_{\boldsymbol{w}}\breve{\mathcal{H}}(\boldsymbol{w}, t, \omega)\right)dt + \mathcal{H}_1\left(\boldsymbol{w} + \frac{1}{2}J^{-1}\nabla_{\boldsymbol{w}}\breve{\mathcal{H}}(\boldsymbol{w}, t, \omega)\right) \circ dW(t), \tag{4}$$

where $\omega \in \Omega$ is a sample point in the probability space Ω. Although the closed form of the solution to the SPDE (4) was given in [46], which is a series of multiple stochastic integrals, it is very difficult to obtain the inverted expression of $\mathcal{H}_0$ and $\mathcal{H}_1$ in terms of $\breve{\mathcal{H}}$.

For the considerations above, we propose to extend the HNNs to the stochastic context from the quadrature point of view, aiming at achieving a learning accuracy comparable to or even superior to the stochastic counterpart of corrected SHNNs. To elaborate this

approach, we first consider the following deterministic Hamiltonian system expressed in integral form for $t = T$:

$$\boldsymbol{y}_T = \boldsymbol{y}_0 + \int_0^T J^{-1}\nabla\mathcal{H}(\boldsymbol{y}_t)dt. \tag{5}$$

We use the fully connected neural network $\mathcal{H}_{net}(\boldsymbol{y},\theta)$ to approximate $\mathcal{H}(\boldsymbol{y})$ where θ represents the learnable parameters in the network. We propose to construct the loss function as follows

$$Loss = \frac{1}{N_0}\sum_{i=1}^{N_0}\left\|\boldsymbol{y}_T - \boldsymbol{y}_0 - I(\boldsymbol{y}_0^i,\theta)\right\|_2^2, \tag{6}$$

where N_0 is the number of observed initial values $\boldsymbol{y}_0^i$ $(i = 1,\ldots,N_0)$, and $I_1(\boldsymbol{y}_0^i,\theta)$ is the approximation of $\int_0^T J^{-1}\nabla H_{net}(\boldsymbol{y}_t^i,\theta)dt$ via numerical quadrature, namely

$$\int_0^T J^{-1}\nabla\mathcal{H}_{net}(\boldsymbol{y}_t^i,\theta)dt \approx \sum_{n=0}^{N} A_n J^{-1}\nabla\mathcal{H}_{net}(\boldsymbol{y}_{t_n}^i,\theta) =: I(\boldsymbol{y}_0^i,\theta), \tag{7}$$

with quadrature nodes t_n, weights A_n $(n = 0,\ldots,N)$, and $\boldsymbol{y}_{t_n}^i$ are observations of $\boldsymbol{y}_t$ at time $t = t_n$ starting from the initial value $\boldsymbol{y}_0^i$. For such a loss function, we see that the discrepancy between the network target, i.e., the function that makes the loss equal zero on arbitrary training data [10], and the true Hamiltonian $\mathcal{H}$ depends on the accuracy of the quadrature formula applied, instead of being determined by the Hamilton–Jacobi PDE (2) fulfilled by the network target $\breve{\mathcal{H}}$ and $\mathcal{H}$ in the corrected SHNN method. Therefore, it would be easier to improve the learning accuracy via using quadrature formulae of higher accuracy in the quadrature-based model, than via correcting $\breve{\mathcal{H}}_{net}$ based on the relation between $\breve{\mathcal{H}}$ and $\mathcal{H}$ that is represented in a series form requiring complex calculations and finding inverse mappings in the corrected SHNN method. Comparisons between the two approaches will be conducted in the numerical experiments in Section 4.

To extend the quadrature-based model to a stochastic context, we still need certain denoising methods, as is explained in the following subsections.

3.2. Neural Network Settings for SHSs

To learn the SHS (1), we parameterize the unknown Hamiltonian functions $\mathcal{H}_0(\boldsymbol{y})$ and $\mathcal{H}_1(\boldsymbol{y})$ by

$$\mathcal{H}_{net0}(\boldsymbol{y},\theta_0) = \boldsymbol{K}_2\sigma(\boldsymbol{K}_1\boldsymbol{y} + \boldsymbol{b}_1) + \boldsymbol{b}_2, \tag{8}$$

and

$$\mathcal{H}_{net1}(\boldsymbol{y},\theta_1) = \boldsymbol{K}_4\sigma(\boldsymbol{K}_3\boldsymbol{y} + \boldsymbol{b}_3) + \boldsymbol{b}_4, \tag{9}$$

respectively, where $\theta_0 = \{\boldsymbol{K}_1,\boldsymbol{K}_2,\boldsymbol{b}_1,\boldsymbol{b}_2\}$, $\theta_1 = \{\boldsymbol{K}_3,\boldsymbol{K}_4,\boldsymbol{b}_3,\boldsymbol{b}_4\}$ are network learnable parameters, and $\sigma(\cdot)$ is the activation function. We denote

$$\begin{aligned} f_{net}(\boldsymbol{y}_t,\theta) &= J^{-1}\nabla\mathcal{H}_{net0}(\boldsymbol{y}_t,\theta_0) + \frac{1}{2}J^{-1}\nabla^2\mathcal{H}_{net1}(\boldsymbol{y}_t,\theta_1)J^{-1}\nabla\mathcal{H}_{net1}(\boldsymbol{y}_t,\theta_1),\\ g_{net}(\boldsymbol{y}_t,\theta) &= J^{-1}\nabla\mathcal{H}_{net1}(\boldsymbol{y}_t,\theta_1), \end{aligned} \tag{10}$$

with $\theta = (\theta_0,\theta_1)$. The solution of SHS (1) can be written in the following integral form for $t = T$

$$\begin{aligned} \boldsymbol{y}_T = \boldsymbol{y}_0 &+ \int_0^T J^{-1}\nabla\mathcal{H}_0(\boldsymbol{y}_t)dt + \int_0^T \frac{1}{2}J^{-1}\nabla^2\mathcal{H}_1(\boldsymbol{y}_t)J^{-1}\nabla\mathcal{H}_1(\boldsymbol{y}_t)dt\\ &+ \int_0^T J^{-1}\nabla\mathcal{H}_1(\boldsymbol{y}_t)dW(t), \end{aligned} \tag{11}$$

where, similar to (10), we write

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{y}) &= J^{-1}\nabla\mathcal{H}_0(\boldsymbol{y}) + J^{-1}\nabla^2\mathcal{H}_1(\boldsymbol{y})J^{-1}\nabla\mathcal{H}_1(\boldsymbol{y}), \\ \boldsymbol{g}(\boldsymbol{y}) &= J^{-1}\nabla\mathcal{H}_1(\boldsymbol{y})). \end{aligned} \tag{12}$$

In order to construct loss functions according to the Equation (11), we denoise it by moments calculations. For a given fixed $\boldsymbol{y}_0$, taking expectations on (11) and using the *Itô* isometry we obtain

$$E(\boldsymbol{y}_T|\boldsymbol{y}_0) = \boldsymbol{y}_0 + E\left(\int_0^T \boldsymbol{f}(\boldsymbol{y}_t)dt|\boldsymbol{y}_0\right), \tag{13}$$

and

$$COV\left(\boldsymbol{y}_T - \boldsymbol{y}_0 - \int_0^T \boldsymbol{f}(\boldsymbol{y}_t)dt\right) = E\left(\int_0^T \boldsymbol{g}(\boldsymbol{y}_t)\boldsymbol{g}(\boldsymbol{y}_t)^T dt|\boldsymbol{y}_0\right). \tag{14}$$

Then, we build loss functions according to the equalities (13) and (14), that is,

$$Loss_1 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left\|E(\boldsymbol{y}_T^i|\boldsymbol{y}_0^i) - \boldsymbol{y}_0^i - I_1(\boldsymbol{y}_0^i,\theta)\right\|_2^2, \tag{15}$$

$$Loss_2 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left\|I_2(\boldsymbol{y}_0^i,\theta) - I_3(\boldsymbol{y}_0^i,\theta)\right\|_2^2, \tag{16}$$

where N_0 is the number of observed initial values $\boldsymbol{y}_0^i$ ($i = 1,\ldots,N_0$), and $I_1(\boldsymbol{y}_0^i,\theta)$ is the approximation of $E\left(\int_0^T \boldsymbol{f}_{net}(\boldsymbol{y}_t^i,\theta)dt|\boldsymbol{y}_0^i\right)$ via numerical quadrature, namely

$$E\left(\int_0^T \boldsymbol{f}_{net}(\boldsymbol{y}_t^i,\theta)dt|\boldsymbol{y}_0^i\right) \approx \sum_{n=0}^{N} A_n E\left(\boldsymbol{f}_{net}(\boldsymbol{y}_{t_n}^i,\theta)|\boldsymbol{y}_0^i\right) =: I_1(\boldsymbol{y}_0^i,\theta), \tag{17}$$

with quadrature nodes t_n, weights A_n (n = 0, …, N), and $\boldsymbol{y}_{t_n}^i$ are observations of $\boldsymbol{y}_t$ at time $t = t_n$ starting from the initial value $\boldsymbol{y}_0^i$. $I_2(\boldsymbol{y}_0^i,\theta)$ is the approximation of the covariance $COV\left(\boldsymbol{y}_T^i - \boldsymbol{y}_0^i - \int_0^T \boldsymbol{f}_{net}(\boldsymbol{y}_t^i,\theta)dt\right)$. For the covariance calculation, there arises the double integral:

$$\begin{aligned} &COV\left(\boldsymbol{y}_T^i - \boldsymbol{y}_0^i - \int_0^T \boldsymbol{f}_{net}(\boldsymbol{y}_t^i,\theta)dt\right) \\ &= E\left(\int_0^T\left(\frac{\boldsymbol{y}_T^i}{T} - \frac{\boldsymbol{y}_0^i}{T} - \boldsymbol{f}_{net}(\boldsymbol{y}_t^i,\theta)\right)dt\int_0^T\left(\frac{\boldsymbol{y}_T^i}{T} - \frac{\boldsymbol{y}_0^i}{T} - \boldsymbol{f}_{net}(\boldsymbol{y}_s^i,\theta)\right)^T ds\right) \\ &= \int_0^T\int_0^T E\left(\left(\frac{\boldsymbol{y}_T^i}{T} - \frac{\boldsymbol{y}_0^i}{T} - \boldsymbol{f}_{net}(\boldsymbol{y}_t^i,\theta)\right)\left(\frac{\boldsymbol{y}_T^i}{T} - \frac{\boldsymbol{y}_0^i}{T} - \boldsymbol{f}_{net}(\boldsymbol{y}_s^i,\theta)\right)^T\right)dsdt, \end{aligned} \tag{18}$$

for which we can derive the corresponding quadrature approximation

$$\begin{aligned} &COV\left(\boldsymbol{y}_T^i - \boldsymbol{y}_0^i - \int_0^T \boldsymbol{f}_{net}(\boldsymbol{y}_t^i,\theta)dt\right) \\ &\approx \sum_{n=0}^{N}\sum_{k=0}^{N} A_n A_k E\left(\left(\frac{\boldsymbol{y}_T^i}{T} - \frac{\boldsymbol{y}_0^i}{T} - \boldsymbol{f}_{net}(\boldsymbol{y}_{t_n}^i,\theta)\right)\left(\frac{\boldsymbol{y}_T^i}{T} - \frac{\boldsymbol{y}_0^i}{T} - \boldsymbol{f}_{net}(\boldsymbol{y}_{t_k}^i,\theta)\right)^T\right) =: I_2(\boldsymbol{y}_0^i,\theta). \end{aligned} \tag{19}$$

$I_3(\boldsymbol{y}_0^i, \theta)$ is the approximation of $E\left(\int_0^T \boldsymbol{g}_{net}(\boldsymbol{y}_t^i, \theta)\boldsymbol{g}_{net}(\boldsymbol{y}_t^i, \theta)^T dt | \boldsymbol{y}_0^i\right)$. Similar to (17)

$$E\left(\int_0^T \boldsymbol{g}_{net}(\boldsymbol{y}_t^i, \theta)\boldsymbol{g}_{net}(\boldsymbol{y}_t^i, \theta)^T dt | \boldsymbol{y}_0^i\right) \approx \sum_{n=0}^{N} A_n E(\boldsymbol{g}_{net}(\boldsymbol{y}_{t_n}^i, \theta)\boldsymbol{g}_{net}(\boldsymbol{y}_{t_n}^i, \theta)^T | \boldsymbol{y}_0^i) =: I_3(\boldsymbol{y}_0^i, \theta). \tag{20}$$

The expectation E in the loss functions can be approximated by taking the mean of M samples, i.e.,

$$E(\boldsymbol{f}_{net}(\boldsymbol{y}_{t_n}^i, \theta)|\boldsymbol{y}_0^i) \approx \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{f}_{net}(\boldsymbol{y}_{t_n}^{i,j}, \theta), \tag{21}$$

where $\boldsymbol{y}_{t_n}^{i,j}$ $(j = 1, \ldots, M)$ is the observation of $\boldsymbol{y}_t$ at time t_n on the j-th sample path starting from $\boldsymbol{y}_0^i$. That is to say, there are M observed sample paths starting from each $\boldsymbol{y}_0^i$ $(i = 1, \ldots, N_0)$ to comprise the data set

$$\mathcal{D} = \left\{\left(\boldsymbol{y}_0^i, \boldsymbol{y}_{t_0}^{i,j}, \ldots, \boldsymbol{y}_{t_N}^{i,j}, \boldsymbol{y}_T^{i,j}\right) | i = 1, \ldots, N_0, j = 1, \ldots, M\right\}, \tag{22}$$

and then $E(\boldsymbol{y}_T^i|\boldsymbol{y}_0^i)$ in (15) can be approximated by

$$E(\boldsymbol{y}_T^i|\boldsymbol{y}_0^i) \approx \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{y}_T^{i,j}. \tag{23}$$

3.3. Training Algorithm

The learning process is summarized in the Algorithm 1.

Algorithm 1: Process of learning drift and diffusion Hamiltonian functions.

Input: Numerical quadrature formula with weights $\{A_n\}_{n=0}^N$ and nodes $\{t_n\}_{n=0}^N$, trajectory data set of SHS $\mathcal{D} = \{(\boldsymbol{y}_0^i, \boldsymbol{y}_{t_0}^{i,j}, ..., \boldsymbol{y}_{t_n}^{i,j}, \boldsymbol{y}_T^{i,j}) | i = 1, \ldots, N_0, j = 1, \ldots, M\}$.

Parameters: $\mathcal{H}_{net0}$ with training parameters θ_0 and $\mathcal{H}_{net1}$ with training parameters θ_1; learning rate l_r, number of training epochs n_{epoch}.

Output: $\mathcal{H}_{net0}$ and $\mathcal{H}_{net1}$.

for $l = 1 : n_{epoch}$ **do**

$I_1(\boldsymbol{y}_0^i, \theta) \leftarrow \sum_{n=0}^{N} A_n E\left(\boldsymbol{f}_{net}(\boldsymbol{y}_{t_n}^i, \theta)|\boldsymbol{y}_0^i\right)$;

$Loss_1 \leftarrow \frac{1}{N_0}\sum_{i=1}^{N_0} \left\|E(\boldsymbol{y}_T^i|\boldsymbol{y}_0^i) - \boldsymbol{y}_0^i - I_1(\boldsymbol{y}_0^i, \theta)\right\|_2^2$;

$I_2(\boldsymbol{y}_0^i, \theta) \leftarrow \sum_{n,k=0}^{N} A_n A_k E\left(\left(\frac{\boldsymbol{y}_T^i}{T} - \frac{\boldsymbol{y}_0^i}{T} - \boldsymbol{f}_{net}(\boldsymbol{y}_{t_n}^i, \theta)\right)\left(\frac{\boldsymbol{y}_T^i}{T} - \frac{\boldsymbol{y}_0^i}{T} - \boldsymbol{f}_{net}(\boldsymbol{y}_{t_k}^i, \theta)\right)^T\right)$;

$I_3(\boldsymbol{y}_0^i, \theta) \leftarrow \sum_{n=0}^{N} A_n E\left(\boldsymbol{g}_{net}(\boldsymbol{y}_{t_n}^i, \theta)\boldsymbol{g}_{net}(\boldsymbol{y}_{t_n}^i, \theta)^T\right)$;

$Loss_2 \leftarrow \frac{1}{N_0}\sum_{i=1}^{N_0} \left\|I_2(\boldsymbol{y}_0^i, \theta) - I_3(\boldsymbol{y}_0^i, \theta)\right\|_2^2$;

$Loss \leftarrow Loss_1 + Loss_2$;

optimize *Loss* by *Adam* and update θ_0 and θ_1;

end

4. Numerical Experiments

The experiments in this section are conducted on a system running Windows 11, and within a virtual environment including Python 3.9.7 and Torch 1.12.0, which is created by using Anaconda.

4.1. Example 1

This example is aimed to compare the learning accuracy, as well as the prediction ability between the corrected SHNN method [14] and our quadrature-based learning model. The underlying Hamiltonian system is the harmonic oscillator

$$d\boldsymbol{y}_t = J^{-1}\nabla\mathcal{H}(\boldsymbol{y})dt, \quad \boldsymbol{y}(0) = \boldsymbol{y}_0, \tag{24}$$

where $\boldsymbol{y} = (p, q)^T$, and $\mathcal{H}(\boldsymbol{y}) = \frac{1}{2}(p^2 + q^2)$.

We denote the fully connected neural network that approximates the Hamiltonian in the quadrature-based model by $\mathcal{H}_{net}$, and the one in the corrected SHNN model by $\breve{\mathcal{H}}_{net}$. Both $\mathcal{H}_{net}$ and $\breve{\mathcal{H}}_{net}$ have a single layer of width 16, utilizing $tanh(\cdot)$ as the activation function. We use Adam with learning rate $l_r = 0.001$ as the optimizer and take $epoch = 50000$. For training, we set $t \in [0, T]$, the number of initial values $N_0 = 1200$, and the initial points $\{\boldsymbol{y}_0^i\}_{i=1}^{1200}$ are uniformly sampled from the region $D_0 = [-4, 4] \times [-4, 4]$. For the quadrature-based model, we choose three different quadrature formulae: the Simpson's rule, the two-point Gaussian quadrature, and the three-point Gaussian quadrature. The weights and nodes for them on the interval $t \in [0, T]$ are listed in the Table 1.

Table 1. Nodes and weights of three different quadrature formulae.

	Simpson	**Two-Point Gaussian**	**Three-Point Gaussian**
Nodes	$\{0, \frac{1}{2}, 1\} \times T$	$\{\frac{1}{2} - \frac{1}{2\sqrt{3}}, \frac{1}{2} + \frac{1}{2\sqrt{3}}\} \times T$	$\{\frac{1}{2} - \frac{\sqrt{15}}{10}, \frac{1}{2}, \frac{1}{2} + \frac{\sqrt{15}}{10}\} \times T$
Weights	$\{\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\} \times T$	$\{\frac{1}{2}, \frac{1}{2}\} \times T$	$\{\frac{5}{18}, \frac{4}{9}, \frac{5}{18}\} \times T$

Denoting the above six different nodes other than $t_0 = 0$ in increasing order by $t_1, \ldots, t_6$, respectively, we generate the training data set

$$\mathcal{D}_0 = \left\{ \left(\boldsymbol{y}_0^i, \boldsymbol{y}_{t_1}^i, \boldsymbol{y}_{t_2}^i, \boldsymbol{y}_{t_3}^i, \boldsymbol{y}_{t_4}^i, \boldsymbol{y}_{t_5}^i, \boldsymbol{y}_{t_6}^i, \boldsymbol{y}_T^i \right) | i = 1, \ldots, 1200 \right\}$$

by using numerical methods on the harmonic oscillator (24) with tiny time-step 0.001, starting from each sampled initial value $\boldsymbol{y}_0^i$ $(i = 1, \ldots, 1200)$.

The loss function of the corrected SHNN model by using the midpoint method as the symplectic integrator is

$$Loss_{SHNN} = \frac{1}{N_0} \sum_{i=1}^{N_0} \left\| \frac{\boldsymbol{y}_{t_3}^i - \boldsymbol{y}_0^i}{h} - J^{-1}\nabla\breve{\mathcal{H}}_{net}\left(\frac{\boldsymbol{y}_{t_3}^i + \boldsymbol{y}_0^i}{2}\right) \right\|_2^2,$$

where we choose $t_0 = 0$ and t_3 as the observation time pair in the corrected SHNN model whereby $h = t_3 - t_0$. After training, $\breve{\mathcal{H}}_{net}$ needs to be corrected according to the following relation [14]

$$\mathcal{H} = \breve{\mathcal{H}} - \frac{h^2}{24}\nabla^2\breve{\mathcal{H}}\left(J^{-1}\nabla\breve{\mathcal{H}}, J^{-1}\nabla\breve{\mathcal{H}}\right) + \mathcal{O}\left(h^4\right)$$

to make $\breve{\mathcal{H}}_{net}$ closer to the true Hamiltonian $\mathcal{H}$.

In this example, we use the following average learning error ε_H defined in [14] to measure the accuracy of approximating H by the learned H_{net} in the corrected SHNN method and our quadrature-based learning model:

$$\varepsilon_H = \langle | H_{net} - H - \langle H_{net} - H \rangle_V | \rangle_V, \tag{25}$$

where the term $\langle f \rangle_V$ denotes the average value of the function f over the region V. This error representation aims to eliminate the additional constant arising from deriving the Hamiltonians from their gradients in the learning process. In our numerical experiment, we take $V = [-3, 3] \times [-3, 3]$.

The Table 2 lists the average errors of learning $\mathcal{H}$, resulting from using the quadrature-based model with three different quadrature formulae, as well as from the corrected SHNN method:

Table 2. Average learning errors by different quadrature formulae and corrected SHNN.

	Corrected SHNN	Simpson	Two-Point Gaussian	Three-Point Gaussian
$\varepsilon_{\mathcal{H}}(h=0.8)$	4.3×10^{-3}	1.8×10^{-3}	1.4×10^{-3}	1.3×10^{-3}
$\varepsilon_{\mathcal{H}}(h=0.4)$	7.1×10^{-4}	5.0×10^{-4}	3.9×10^{-4}	3.0×10^{-4}

Hereby, we have calculated the learning errors by $h=0.4$ and $h=0.8$, where h is the time step used in $Loss_{SHNN}$. It can be seen that the three quadrature-based models provide more accurate $\mathcal{H}$ than the corrected SHNN method, and the learning accuracy increases with using more precise quadrature formulae in the quadrature-based models.

Figure 1 compares the evolution of $p(t)$ on $[0,200]$ predicted by the corrected SHNN model and three quadrature-based models with the true solution, for $h=0.8$ and $\boldsymbol{y}_0=(0.2,0.2)^T$. Figure 2 compares predictions of $p(t)$ on $t\in[0,400]$ made by the corrected SHNN method and the three-point Gaussian quadrature-based model with that of the true solution, for $h=0.4$ and $\boldsymbol{y}_0=(1,0)^T$. It is clear that the quadrature-based models behave better in long-term predictions than the corrected SHNN, and the three-point Gaussian quadrature provides the best learning accuracy among the three quadrature formulae.

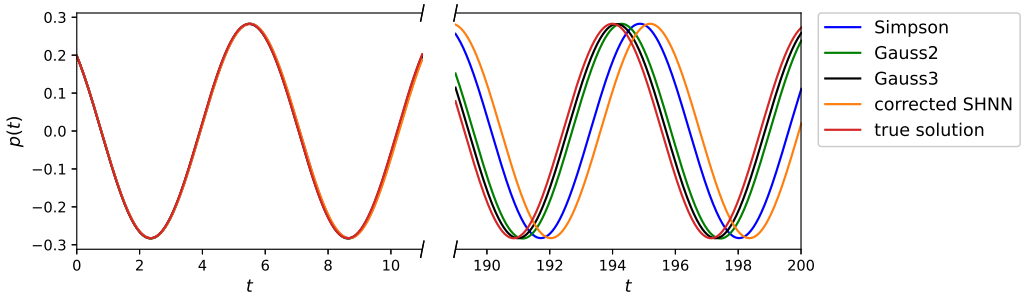

Figure 1. Predicted trajectories of $p(t)$ by different learning methods, with $h=0.8$, $\boldsymbol{y}_0=(0.2,0.2)^T$.

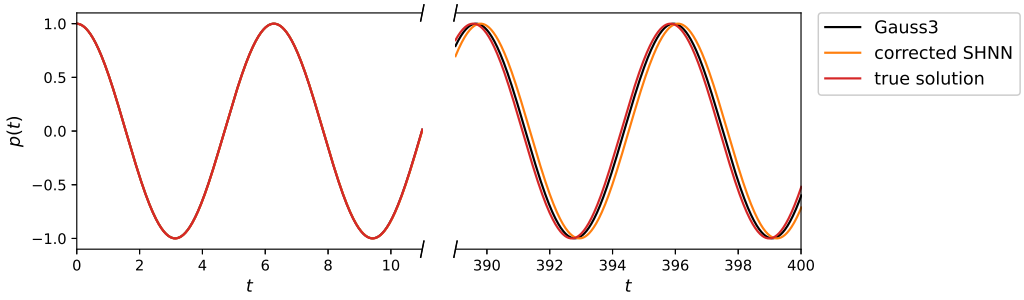

Figure 2. Predicted trajectories of $p(t)$ by different learning methods, with $h=0.4$, $\boldsymbol{y}_0=(1,0)^T$.

4.2. Example 2

In this example, we consider the stochastic Kubo oscillator [2] which is a benchmark stochastic Hamiltonian system with Hamiltonian functions

$$\mathcal{H}_0(p,q)=p^2+q^2,\quad \mathcal{H}_1(p,q)=\frac{3}{20}(p^2+q^2), \tag{26}$$

where $\boldsymbol{y}=(p,q)^T\in\mathbb{R}^2$. For the training we let $T=1$, $M=100$, the number of initial values $N_0=1200$, and the initial points $\{\boldsymbol{y}_0^i\}_{i=1}^{1200}$ are sampled uniformly from the region $D_0=[-4,4]\times[-4,4]$. We generate the training data set

$$\mathcal{D}_1 = \left\{ \left(\boldsymbol{y}_0^i, \boldsymbol{y}_{t_1}^{i,j}, \boldsymbol{y}_{t_2}^{i,j}, \boldsymbol{y}_{t_3}^{i,j}, \boldsymbol{y}_{t_4}^{i,j}, \boldsymbol{y}_{t_5}^{i,j}, \boldsymbol{y}_{t_6}^{i,j}, \boldsymbol{y}_1^{i,j} \right) | i = 1, \ldots, 1200, \ j = 1, \ldots, 100 \right\} \tag{27}$$

by using numerical methods on the Kubo oscillator system with appropriate tiny step sizes, starting from each sampled $\boldsymbol{y}_0^i$ $(i = 1, \ldots, 1200)$.

The fully connected layers $\boldsymbol{K}_1$ and $\boldsymbol{K}_3$ in the network are of size $N_{width} \times N_{input} = 16 \times 2$ and initialized as $\boldsymbol{K}_1, \boldsymbol{K}_3 \sim \mathcal{N}\left(0, \sqrt{2/\left[N_{width} \times N_{input}\right]}\right)$. The fully connected layers $\boldsymbol{K}_2$ and $\boldsymbol{K}_4$ are initialized as $\boldsymbol{K}_2, \boldsymbol{K}_4 \sim \mathcal{N}\left(0, \sqrt{2/\left[N_{width} \times N_{output}\right]}\right)$ with size $N_{output} \times N_{width} = 1 \times 16$. $\boldsymbol{b}_1$ and $\boldsymbol{b}_3$ are $N_{width} = 16$-dimensional bias and initialized as $\boldsymbol{b}_1, \boldsymbol{b}_3 \sim \mathcal{N}\left(0, \sqrt{2/N_{width}}\right)$. $\boldsymbol{b}_2$ and $\boldsymbol{b}_4$ are $N_{output} = 1$-dimensional bias and initialized as $\boldsymbol{b}_2, \boldsymbol{b}_4 \sim \mathcal{N}\left(0, \sqrt{2/N_{output}}\right)$. $tanh(x)$ is chosen as the activation function, and we train the network with $epoch = 20000$.

The three panels in Figure 3, from left to right, depict the comparison in the phase space between the true solutions and the predicted solutions generated by the neural network using the Simpson quadrature, the two-point and the three-point Gaussian quadrature, respectively. Hereby. we take $t \in [0, 10]$, and both the predicted and the true solutions are obtained by applying the midpoint scheme to the learned and the true system, respectively, with time step $h_{prediction} = 0.01$ for prediction and time step $h_{true} = 0.001$ for simulating the true solution, accordingly. It can be seen that the three-point Gaussian quadrature implies the most accurate prediction.

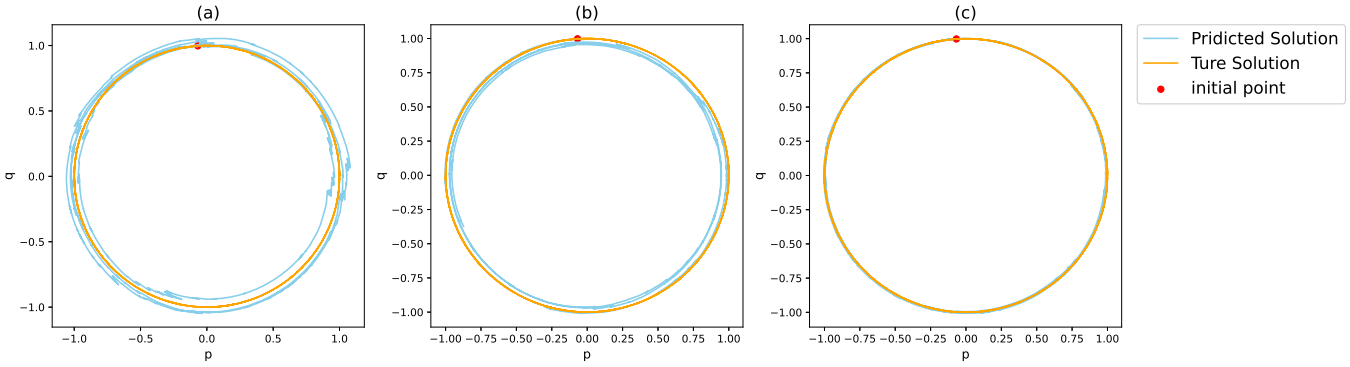

Figure 3. Predicted and true solutions in the phase space. (**a**–**c**) are the true solution in the phase space compared to predicted solution based on the Simpson, two-point, and three-point Gaussian quadrature, respectively.

Figure 4 compares the evolution of the predicted and true $p(t)$ and $q(t)$ on $t \in [0, 10]$, whereby the network predictions in the left, middle and right panels are based on the Simpson, the two-point and the three-point Gaussian quadrature, respectively. Again, the prediction accuracy increases from left to right.

Figure 5 visualizes the learning error distribution of $\mathcal{H}_0$ (panels (a1), (b1), (c1)) and $\mathcal{H}_1$ (panels (a2), (b2), (c2)) in the phase space, whereby the results in panels (ai), (bi), and (ci) $(i = 1, 2)$ are obtained using the Simpson, the two-point and the three-point Gauss quadrature, respectively. Here, the error function between the learned Hamiltonian $\mathcal{H}_{\text{net}}(p, q)$ and the true Hamiltonian $\mathcal{H}(p, q)$ is defined as [14]:

$$\varepsilon_{\mathcal{H}}(p, q) = |\mathcal{H}_{net}(p, q) - \mathcal{H}(p, q) - \langle \mathcal{H}_{net} - \mathcal{H} \rangle_V|, \tag{28}$$

which is a function of $\boldsymbol{y} = (p, q)^T$ and equals the average learning error (25) after taking average over the region V. In this experiment, we take $V = [-1, 1] \times [-1, 1]$. It can be seen from the figures that the learning errors of the Hamiltonian functions are distributed in an almost uniform way in the phase space, meaning that there are no particular regions

exhibiting significantly large or small errors. Meanwhile, it is clear that utilizing the two- and three-point Gaussian quadrature can produce better learning accuracy.

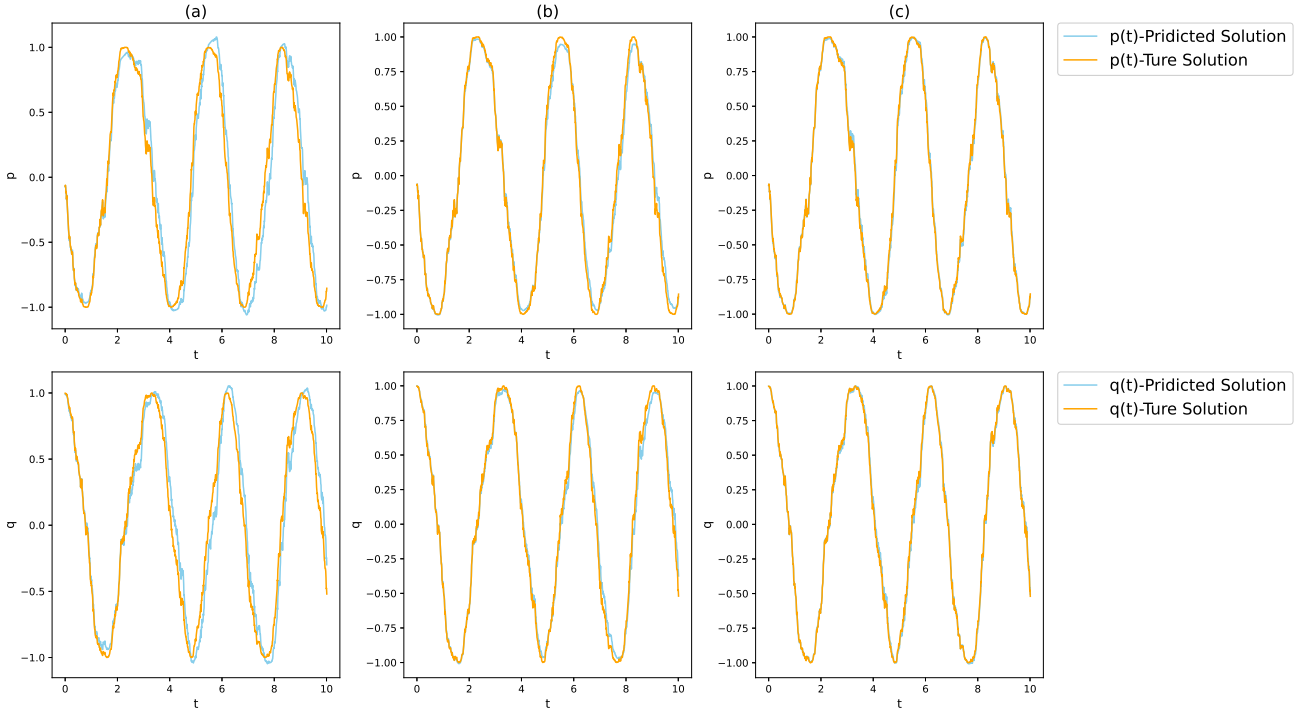

Figure 4. Evolution of the predicted and true solutions. (**a–c**) are the true solution of $p(t)$ and $q(t)$ compared to predicted solution based on the Simpson, two-point, and three-point Gaussian quadrature, respectively.

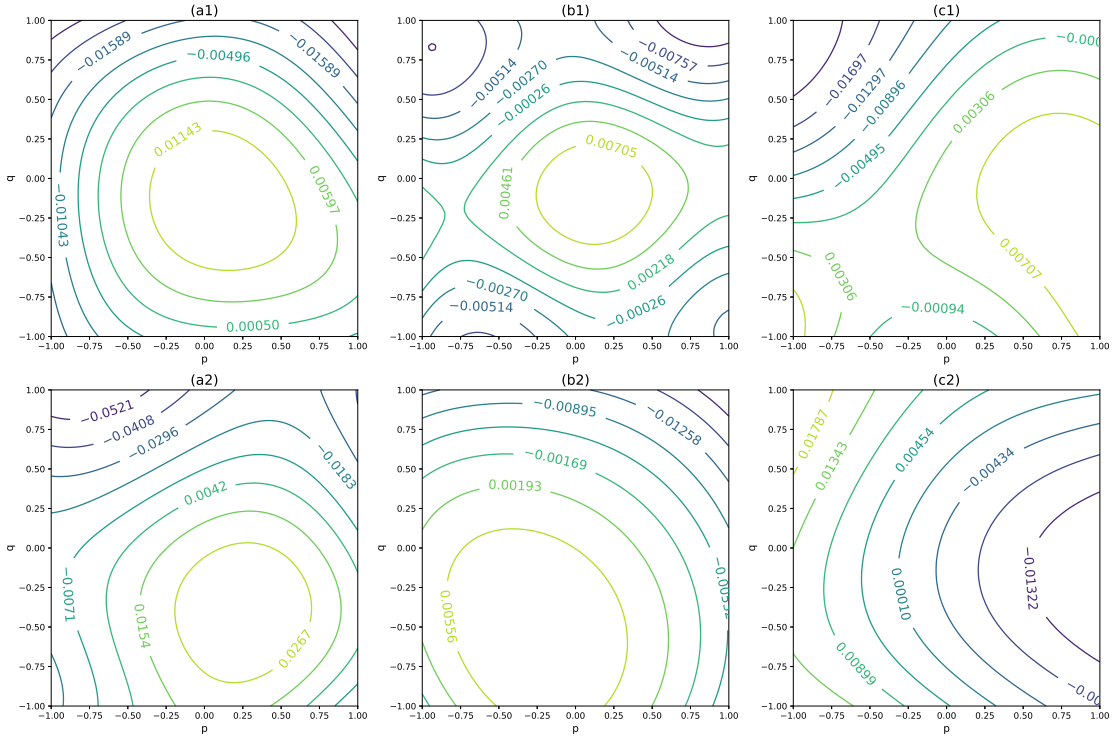

Figure 5. Learning error distribution in the phase space. (**a1–c1**) are the error distributions for $\mathcal{H}_0$. (**a2–c2**) are the error distributions for $\mathcal{H}_1$. (**a***i*), (**b***i*), and (**c***i*) $(i = 1, 2)$ are obtained using the Simpson, the two-point and the three-point Gauss quadrature, respectively.

The average learning error (25) on the region $V = [-1,1] \times [-1,1]$ for $\mathcal{H}_{net0}$ and $\mathcal{H}_{net1}$ by using the three different quadrature formulae are listed in the following table:

The errors reported in Table 3 are the average results from ten tests. It is evident that the three-point Gaussian quadrature yields the best accuracy, while the error by Simpson's formula is the largest among the three.

Table 3. Average learning errors by different quadrature formulae.

	Simpson	**Two-Point Gaussian**	**Three-Point Gaussian**
$\varepsilon_{\mathcal{H}_0}$	0.0190	0.0045	0.0041
$\varepsilon_{\mathcal{H}_1}$	0.0170	0.0070	0.0054

Now we compare our method with that in [48], the idea of which is as follows. For the stochastic differential equations with learnable parameters θ

$$dx_t = f(x_t, \theta)dt + \sigma(x_t, \theta)dW_t, \tag{29}$$

the following Kramers–Moyal formula holds:

$$\begin{aligned} f(x_t, \theta) &= \lim_{\Delta t \to 0} E\left[\frac{X_{t+\Delta t} - X_t}{\Delta t} | X_t = x_t\right], \\ \sigma^2(x_t, \theta) &= \lim_{\Delta t \to 0} E\left[\frac{(X_{t+\Delta t} - X_t)^2}{\Delta t} | X_t = x_t\right]. \end{aligned} \tag{30}$$

Based on (30), the loss functions are built as

$$\begin{aligned} loss_{drift} &= \sum_{i=0}^{N_0} \left\| f(x_0^i, \theta) - \sum_{j=1}^{\hat{N}} \left[\frac{x_1^{i,j} - x_0^i}{\Delta t}\right] \right\|^2, \\ loss_{diffusion} &= \sum_{i=0}^{N_0} \left\| \sigma^2(x_t, \theta) - \sum_{j=1}^{\hat{N}} \left[\frac{(x_1^{i,j} - x_0^i)^2}{\Delta t}\right] \right\|^2, \end{aligned} \tag{31}$$

where we take $N_0 = 1200$, $\hat{N} = 500$ in our experiments, and $\{(x_0^i, x_1^{i,j}) | i = 1, \ldots, N_0, j = 1, \ldots, \hat{N}\}$ is the training data set. $\{x_1^{i,j}\}_{j=1}^{\hat{N}}$ are numerically generated from the initial point x_0^i with time step Δt and the initial points $\{x_0^i\}_{i=1}^{N_0}$ are uniformly sampled from the region $D_0 = [-4,4] \times [-4,4]$. We take $\Delta t = 0.01$. Both $f(x, \theta)$ and $\sigma(x, \theta)$ are single-layer, fully connected neural networks with a width of 32. The training epoch is 1000, and the learning rate of Adam is 0.001. For convenience, we will refer to the above method of learning SDEs as the Kramers–Moyal method. Next, we compare the predicted solution obtained by the Kramers–Moyal method with that of our quadrature model.

Figure 6 draws the predicted $(p(t), q(t))$ on $[0, 10]$ arising from the two learning methods, as well as from the true system. It is obvious that the three-point Gaussian-based model provides better learning accuracy and possesses stronger generalization ability than the Kramers–Moyal method.

4.3. Example 3

In this subsection, we consider the stochastic Hamiltonian system with Hamiltonians:

$$\mathcal{H}_0(p,q) = \frac{1}{2}p^2 - 4cos(q), \quad \mathcal{H}_1(p,q) = \frac{1}{2}q + 0.2sin(q), \tag{32}$$

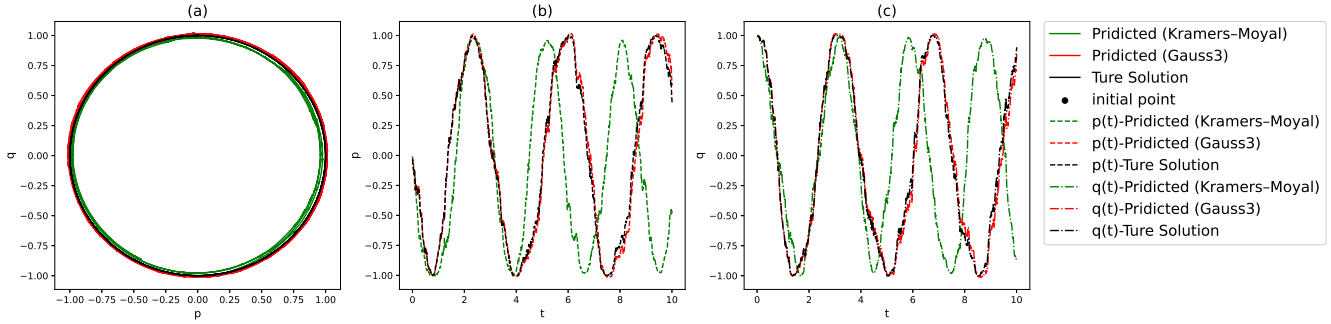

Figure 6. Predicted $(p(t), q(t))$ on $[0, 10]$ by the Kramers–Moyal method and the quadrature based model. (**a**) is the comparison in phase space. (**b**) is a comparison of $p(t)$ and (**c**) is a comparison of $q(t)$.

The setup of training data and neural network is the same as in Example 2.

Figure 7 displays the comparison between $\frac{\partial \mathcal{H}_i}{\partial q}$ and $\frac{\mathcal{H}_{neti}}{\partial q}$ $(i = 0, 1)$, by using the three mentioned quadrature in the learning, where the panels from left to right correspond to Simpson's rule, the two-point and three-point Gauss quadrature formula, respectively.

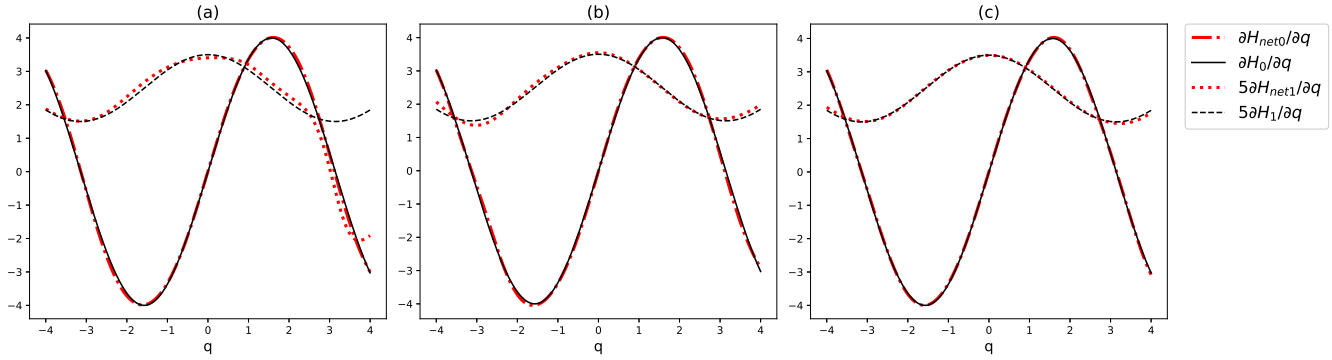

Figure 7. Comparison of $\frac{\partial \mathcal{H}}{\partial q}$ resulted from different quadrature formulae. (**a**–**c**) are resulted from the Simpson, two-point, and three-point Gaussian quadrature, respectively.

The left, middle and right panels in Figure 8 illustrate the comparison between the true and predicted phase trajectories generated by the neural network using the Simpson's rule, two-point and three-point Gaussian quadrature, respectively, on $t \in [0, 10]$. The predicted and true solutions are simulated by the midpoint discretization on the learned and true system, respectively, with time step $h_{prediction} = 0.01$ for prediction, and $h_{true} = 0.001$ for the true solution. Obviously, the Gaussian quadrature formulae provide better prediction than Simpson's rule, and the three-point Gaussian quadrature behaves the best.

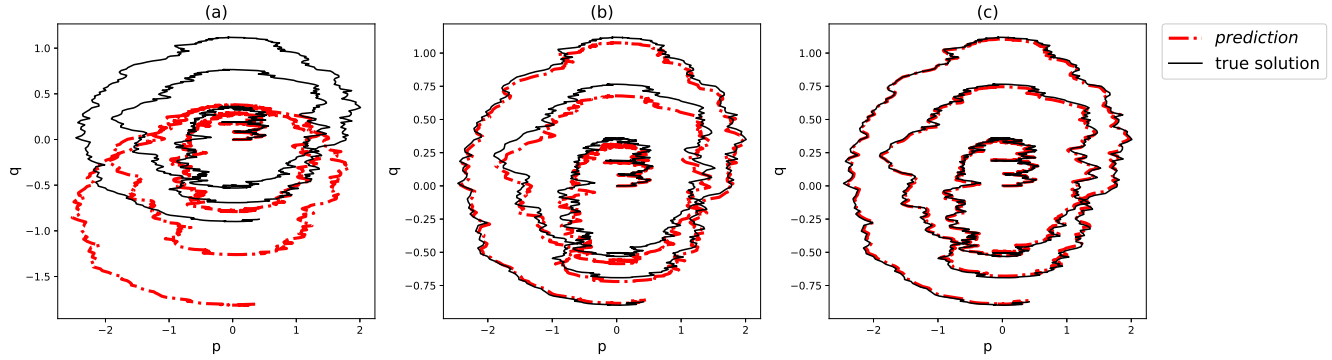

Figure 8. Predicted and true solutions in the phase space. (**a**–**c**) are the true solution in the phase space compared to predicted solution based on the Simpson, two-point, and three-point Gaussian quadrature, respectively.

In Figure 9, the spread of the learning error function (28) in the phase space for $\mathcal{H}_{net0}$ resulting from the Simpson's rule, the two-point, and the three-point Gaussian quadrature formulae, are illustrated in the panels (a1), (b1) and (c1), respectively, and those for $\mathcal{H}_{net1}$ are shown in (a2), (b2) and (c2), accordingly.

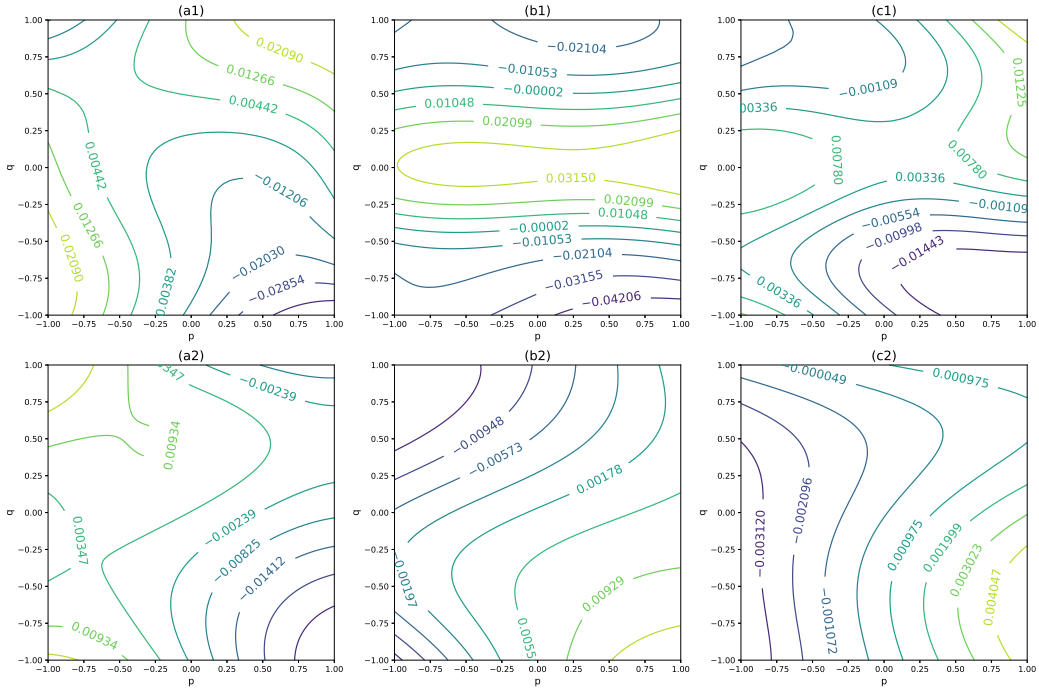

Figure 9. Learning error distribution in the phase space. (**a1–c1**) are the error distributions for $\mathcal{H}_0$. (**a2–c2**) are the error distributions for $\mathcal{H}_1$. (**a***i*), (**b***i*), and (**c***i*) $(i = 1, 2)$ are obtained using the Simpson, the two-point and the three-point Gauss quadrature, respectively.

The average learning error (25) on the region $V = [-1, 1] \times [-1, 1]$ for $\mathcal{H}_{net0}$ and $\mathcal{H}_{net1}$ by using the three different quadrature formulae are listed in the following table:

The errors presented in Table 4 are the mean values from ten tests. The advantage of the Gaussian quadrature formulae, especially the three-point one is evident.

Table 4. Average learning errors by different quadrature formulae.

	Simpson	**Two-Point Gaussian**	**Three-Point Gaussian**
$\varepsilon_{\mathcal{H}_0}$	0.0146	0.0090	0.0043
$\varepsilon_{\mathcal{H}_1}$	0.0095	0.0057	0.0002

4.4. Example 4

In this example, we compare our method with the numerics-informed likelihood method given in [35], which simulates the probability density function of $\boldsymbol{y}$ based on the Euler–Maruyama scheme, and we also compare with the Kramers–Moyal method. The considered stochastic Hamiltonian system is the same as that in Example 3, which can be written explicitly as the Itô (equivalent with Stratonovich) form:

$$dq_t = p_t\, dt, \tag{33}$$

$$dp_t = b(q_t, p_t)\, dt + \sigma(q_t, p_t)\, dW_t, \tag{34}$$

where $\boldsymbol{y} = (p, q)^T$, $b(q_t, p_t) = -4\sin(q_t)$, and $\sigma(q_t, p_t) = -\left(\frac{1}{2} + 0.2\cos(q_t)\right)$. As described in [35], when using the likelihood method, Equation (33) is assumed to be known, while b and σ are unknown.

In the experiment, we build the fully connected neural networks $b_{net}(\boldsymbol{y})$ and $\sigma_{net}(\boldsymbol{y})$ to approximate $b(\boldsymbol{y})$ and $\sigma(\boldsymbol{y})$, respectively. Both $b_{net}(\boldsymbol{y})$ and $\sigma_{net}(\boldsymbol{y})$ have 2 layers each with 50 neurons, and employ the exponential linear unit (ELU) as the activation function. The loss function for training b_{net} and σ_{net} by the likelihood method is

$$Loss = -\frac{1}{N_0}\sum_{i=1}^{N_0}\left(\frac{(p_h^i - p_0^i - b_{net}(q_h^i, p_0^i)h)^2}{2\sigma_{net}((q_h^i, p_0^i))^2} + log|h\sigma_{net}((q_h^i, p_0^i))^2| + log(2\pi)\right),$$

for which we set $h = 0.01$, and the number of initial values $N_0 = 20000$. As in [35], $\{p_0^i\}_{i=1}^{N_0}$ and $\{q_h^i\}_{i=1}^{N_0}$ are regarded as initial observations which we sample uniformly from $V_0 = [-4,4] \times [-4,4]$. The values $\{p_h^i\}_{i=1}^{N_0}$ are generated as in [35] using an approximated adjoint symplectic Euler–Maruyama method applied to (33) and (34) with a tiny step size $h_0 = 0.001$, over 10 steps, where the fixed $\{q_h^i\}_{i=1}^{N_0}$ are applied to approximate $\{q_{k\cdot h_0}^i\}_{i=1}^{N_0}$ (for all $k = 0, \ldots, 9$) appearing in the 10 steps numerical simulation. The training epoch is 200, the learning rate of Adam is 0.01, and the batch size is 32. The setup of training data and neural network for the Kramers–Moyal method is the same as in Example 2. Next, we compare the b_{net} and σ_{net} obtained by the above likelihood method and the Kramers–Moyal method with the results from our method in Example 3.

Figure 10 compares the learned $b_{net}(q)$ and $\sigma_{net}(q)$ by using the likelihood method, the Kramers–Moyal method, the three-point Gaussian quadrature based method, with the true $b(q)$ and $\sigma(q)$, in the left and right panels, respectively.

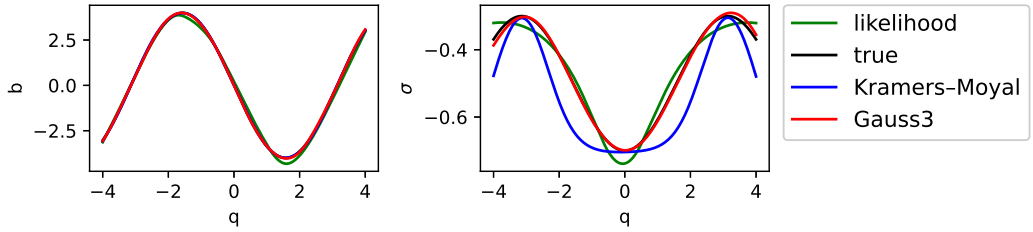

Figure 10. Approximationsof b and σ by the likelihood method, the Kramers–Moyal method and the quadrature-based model.

Figure 11 draws the predicted $q(t)$ on $[0, 20]$ arising from the three learning methods as well as from the true system. It is obvious that the three-point Gaussian-based model provides better learning accuracy and possesses stronger generalization ability than the likelihood method and the Kramers–Moyal method.

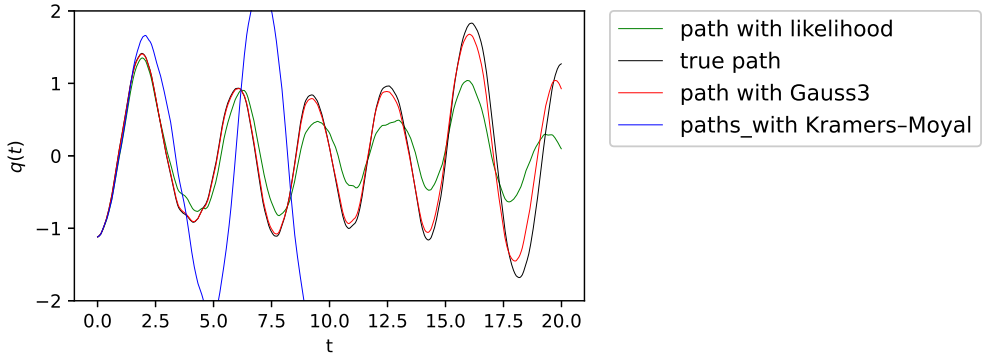

Figure 11. Predicted $q(t)$ on $[0, 20]$ by the likelihood method, the Kramers–Moyal method and the quadrature-based model.

Figure 12 illustrates the probability density functions (PDFs) of $q(t)$ at three time points estimated using 100 trajectories generated from 100 initial points with $q_0 \sim \mathcal{N}(0, 1/10)$, $p_0 = 0$ with respect to the system learned by the likelihood and the quadrature based

model. It can be seen that the three-point Gaussian quadrature-based model produces good estimates of the PDFs for both small and large time points t, while the likelihood method only gives an approximate PDF for a small t (e.g., $t = 0.5$).

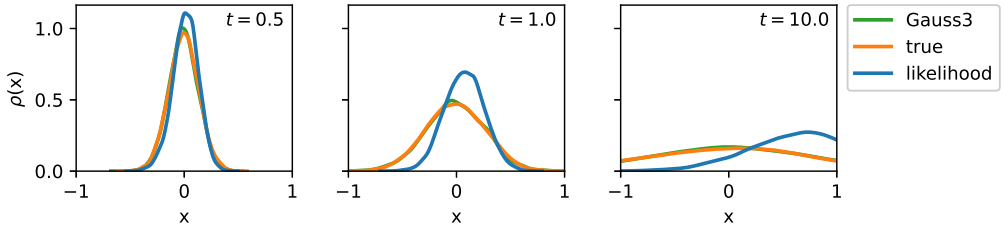

Figure 12. Probability density function $\rho(x)$ of $q(t)$ at times $t = 0.5, 1.0, 10.0$, starting with $q_0 \sim \mathcal{N}(0, 1/10)$, $p_0 = 0$.

4.5. Example 5

In the last example, we consider the system where the Hamiltonian functions are

$$\mathcal{H}_0(p,q) = \frac{p^2 + pq + q}{1 + (p+q)^2}, \quad \mathcal{H}_1(p,q) = 0.1 \ln(1 + p^2 + q^2), \tag{35}$$

on which we compare our method with the Kramers–Moyal method. The configuration of the training data and neural network is the same as in Example 1, except that we take a smaller $T = 0.5$ for the quadrature model. The setup of training data and neural network for the Kramers–Moyal method is the same as in Example 2. Next, we illustrate the predicted solution by the Kramers–Moyal method and our quadrature-based model.

Figure 13 draws the predicted $(p(t), q(t))$ on $[0, 10]$ arising from the two learning methods, as well as from the true system. It is evident that the three-point Gaussian-based model behaves better in accuracy and generalization ability than the Kramers–Moyal method.

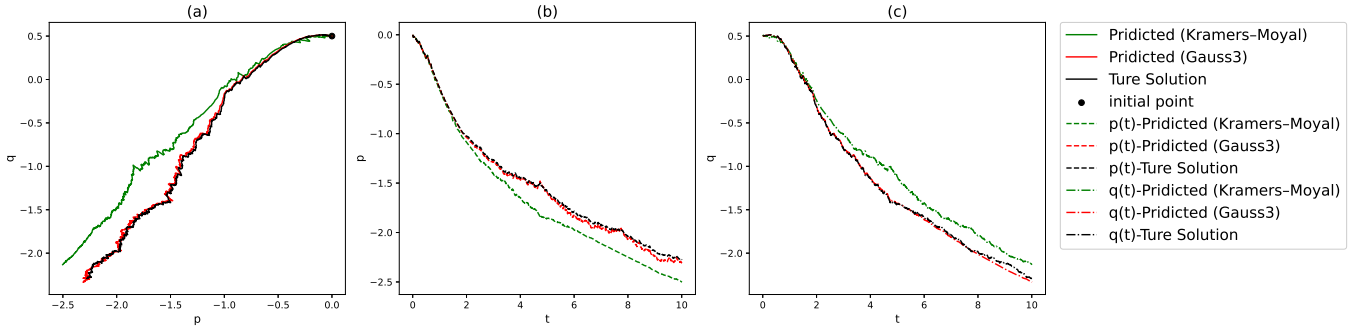

Figure 13. Predicted $(p(t), q(t))$ on $[0, 10]$ by the Kramers–Moyal method and the quadrature based model. (**a**) is the comparison in phase space. (**b**) is a comparison of $p(t)$ and (**c**) is a comparison of $q(t)$.

4.6. Summary

Examples 1, 2, 4 and 5 contribute to comparing our method with the existing SHNN method, the numerics-informed likelihood method and the Kramers–Moyal method applied to deterministic/stochastic harmonic Hamiltonian oscillators as well as the other two SHSs with non-polynomial Hamiltonian functions. Experimental results show the superiority of our method in learning accuracy and long-time prediction, which may essentially result from the quadrature-based model structure and the moment denoising technique. Examples 2 and 3 make a comparison among the applications of three different quadrature formulae in the model, including the Simpson's rule, the two-point and three-point Gaussian quadrature formulae. It is shown that the learning accuracy and the prediction ability of our method increase with employing more precise quadrature formulae.

5. Limitations of the Study

There are still several limitations of the study that should be mentioned. First, a rigorous theoretical convergence analysis is needed for a more insightful observation of the algorithm. Second, we have discussed the case of SHSs with a single noise, while the application of our method to systems with multiple noises may encounter complex calculations in moments of denoising. Third, the utilization of moments calculations necessitates larger datasets than path-wise fitting.

6. Conclusions

We proposed a neural network learning method for detecting the drift and diffusion Hamiltonian functions of stochastic Hamiltonian systems from data, where the loss functions are built upon the integral formulation of the system solution with the integrals being approximated by numerical quadrature formulae, and the stochasticity being 'removed' via moments calculations. Numerical experiments show the effectiveness of the methodology and indicate that the learning accuracy of the proposed models can be improved by employing quadrature formulae of higher accuracy. Moreover, the numerical comparison with the corrected SHNN method, the numerics-informed likelihood method and the Kramers–Moyal method on four concrete Hamiltonian systems show better accuracy and stronger generalization ability of our models. This may mainly be owed to the integral formulation of the loss that enables a more feasible reduction in the learning error via precise quadrature formulae instead of through the complex Hamilton–Jacobi approach by classical models on one hand, and the simple moment fitting method rather than intricate denoising technique on the other hand, which enlarges the possibility of better generalization.

The proposed models generalize the HNNs for deterministic Hamiltonian systems to stochastic context and provide a new way of improving the accuracy and prediction ability of the HNNs from the integral point of view.

Acknowledging the prevalence of multiple noise phenomena in real-world scenarios, our future work will consider expanding the method to SHSs featuring high-dimensional noises. Moreover, the theoretical convergence analysis of the algorithm for a deeper exploration of the inherent nature of the method will be a topic of our further investigation.

Author Contributions: Methodology, X.C., L.W. and Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: The first and second authors are supported by the National Natural Science Foundation of China (NNSFC No.11971458). The third author is supported by the U.S. Department of Energy under the grant number DE-SC0022253.

Data Availability Statement: The data that support the finding of this study are openly available in GitHub at https://github.com/nirynok/Quadrature-model.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Bismut, J.M. Mécanique aléatoire. In *Ecole d'Eté de Probabilités de Saint-Flour X—1980. Lecture Notes in Mathematics*; Hennequin, P.L., Ed.; Springer: Berlin/Heidelberg, Germany, 1982; Volume 929.
2. Milstein, G.N.; Tretyakov, M.V.; Repin, Y.M. Numerical methods for stochastic systems preserving symplectic structure. *SIAM J. Numer. Anal.* **2002**, *40*, 1583–1604. [CrossRef]
3. Wang, L.J. Variational Integrators and Generating Functions for Stochastic Hamiltonian Systems. Ph.D. Thesis, Karlsruhe Institute of Technology, Baden-Württemberg, Germany, 2007.
4. Lewis, J.T.; Maassen, H. Hamiltonian models of classical and quantum stochastic processes. In *Quantum Probability and Applications to the Quantum Theory of Irreversible Processes, Proceedings of the International Workshop, Villa Mondragone, Italy, 6–11 September 1982*; Springer: Berlin/Heidelberg, Germany, 1984; pp. 245–276.
5. Yong, J.; Zhou, X.Y. Maximum Principle and Stochastic Hamiltonian Systems. In *Stochastic Controls: Hamiltonian systems and HJB Equations*; Karatzas, I., Yor, M., Eds.; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1999; pp. 101–153.

6. Seifi, M.; Soltanmanesh, A.; Shafiee, A. Mimicking classical noise in ion channels by quantum decoherence. *Sci. Rep.* **2024**, *14*, 16130. [CrossRef] [PubMed]
7. Chen, T.Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K. Neural ordinary diferential equations. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; pp. 6571–6583.
8. Greydanus, S.; Dzamba, M.; Yosinski, J. Hamiltonian Neural Networks. In Proceedings of the Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
9. Chen, Z.D.; Zhang, J.Y.; Arjovsky, M.; Bottou, L. Symplectic Recurrent Neural Networks. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
10. Zhu, A.Q.; Jin, P.Z.; Tang, Y.F. Deep Hamiltonian networks based on symplectic integrators. *arXiv* **2020**, arXiv:2004.13830.
11. Jin, P.Z.; Zhang, Z.; Zhu, A.Q.; Tang, Y.F.; Karniadakis, G.E. SympNets: Intrinsic structure preserving symplectic networks for identifying Hamiltonian systems. *Neural Netw.* **2020**, *132*, 166–179. [CrossRef] [PubMed]
12. Xiong, S.Y.; Tong, Y.J.; He, X.Z.; Yang, S.Q.; Yang, C.; Zhu, B. Nonseparable Symplectic Neural Networks. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
13. Tong, Y.J.; Xiong, S.Y.; He, X.Z.; Pan, G.H.; Zhu, B. Symplectic neural networks in Taylor series form for Hamiltonian systems. *J. Comput. Phys.* **2021**, *437*, 110325. [CrossRef]
14. David, M.; Méhats, F. Symplectic Learning for Hamiltonian Neural Networks. *J. Comput. Phys.* **2023**, *494*, 112495. [CrossRef]
15. Chen, R.Y.; Tao, M.L. Data-driven prediction of general Hamiltonian dynamics via learning exactly-symplectic maps. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 8–24 July 2021; pp. 1717–1727.
16. Tzen, B.; Raginsky, M. Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit. *arXiv* **2019**, arXiv:1905.09883.
17. Li, X.; Wong, T.L.; Chen, T.Q.; Duvenaud, D. Scalable Gradients and Variational Inference for Stochastic Differential Equations. In Proceedings of the 2nd Symposium on Advances in Approximate Bayesian Inference, Vancouver, BC, Canada, 8 December 2019.
18. Yildiz, C.; Heinonen, M.; Intosalmi, J.; Mannerstrom, H.; Lahdesmaki, H. Learning stochastic differential equations with gaussian processes without gradient matching. In Proceedings of the IEEE 28th International Workshop on Machine Learning for Signal Processing, Aalborg, Denmark, 17–20 September 2018.
19. Jia, J.T.; Benson, A.R. Neural Jump Stochastic Differential Equations. In Proceedings of the Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 9843–9854.
20. Kong, L.K.; Sun, J.M.; Zhang, C. SDE-Net: Equipping Deep Neural Networks with Uncertainty Estimates. In Proceedings of the 37th International Conference on Machine Learning, Online, 13–18 July 2020.
21. Archibald, R.; Bao, F.; Cao, Y.; Sun, H. Numerical analysis for convergence of a sample-wise backpropagation method for training stochastic neural networks. *SIAM J. Numer. Anal.* **2024**, *62*, 593–621. [CrossRef]
22. Gobet, E.; Hoffmann, M.; Reiß, M. Nonparametric Estimation of Scalar Diffusions Based on Low Frequency Data. *Ann. Stat.* **2004**, *32*, 2223–2253. [CrossRef]
23. Song, Y.; Sohl-Dickstein, J.; Kingma, D.P.; Kumar, A.; Ermon, S.; Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv* **2020**, arXiv:2011.13456.
24. Liu, Y.; Yang, M.; Zhang, Z.; Bao, F.; Cao, Y.; Zhang, G. Diffusion-Model-Assisted Supervised Learning of Generative Models for Density Estimation. *JMLMC* **2024**, *5*, 1–13. [CrossRef]
25. Xu, W.; Chen, R.T.; Li, X.; Duvenaud, D.K. Infinitely Deep Bayesian Neural Networks with Stochastic Differential Equations. In Proceedings of the 25th International Conference on Artificial Intelligence and Statistics, Virtual, 28–30 March 2022.
26. Kidger, P.; Foster, J.; Li, X.; Oberhauser, H.; Lyons, T. Neural SDEs as Infinite-Dimensional GANs. In Proceedings of the International Conference on Machine Learning, Online, 18–24 July 2021; pp. 5453–5463.
27. Chen, X.L.; Wang, H.; Duan, J.Q. Detecting stochastic governing laws with observation on stationary distributions. *Phys. D Nonlinear Phenom.* **2023**, *448*, 133691. [CrossRef]
28. Chen, X.L.; Yang, L.; Duan, J.Q.; Karniadakis, G.E. Solving inverse stochastic problems from discrete particle observations using the Fokker–Planck equation and physics informed neural networks. *SIAM J. Sci. Comput.* **2021**, *43*, B811–B830. [CrossRef]
29. Dai, M.; Duan, J.Q.; Hu, J.Y.; Wen, J.H.; Wang, X.J. Variational inference of the drift function for stochastic differential equations driven by Lévy processes. *Chaos* **2022**, *32*, 061103. [CrossRef] [PubMed]
30. Li, Y.; Duan, J.Q. A data-driven approach for discovering stochastic dynamical systems with non-Gaussian Lévy noise. *Phys. D Nonlinear Phenom.* **2021**, *417*, 132830. [CrossRef]
31. Lu, Y.B.; Li, Y.; Duan, J.Q. Extracting stochastic governing laws by non-local Kramers Moyal formulae. *Philos. Trans. R. Soc. A* **2022**, *380*, 20210195. [CrossRef] [PubMed]
32. Solin, A.; Tamir, E.; Verma, P. Scalable inference in SDEs by direct matching of the Fokker–Planck-Kolmogorov equation. In Proceedings of the Conference on Neural Information Processing Systems, Online, 6–14 December 2021.
33. Opper, M. Variational inference for stochastic differential equations. *Ann. Phys.* **2019**, *531*, 1800233. [CrossRef]
34. Ryder, T.; Golightly, A.; McGough, S.; Prangle, D. Black-box variational inference for stochastic differential equations. In Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 4423–4432.

35. Dietrich, F.; Makeev, A.; Kevrekidis, G.; Evangelou, N.; Bertalan, T.; Reich, S.; Kevrekidis, I.G. Learning effective stochastic differential equations from microscopic simulations: Linking stochastic numerics to deep learning. *Chaos* **2023**, *33*, 023121. [CrossRef] [PubMed]
36. Deng, R.Z.; Chang, B.; Brubaker, M.A.; Mori, G.; Lehrmann, A.M. Modeling continuous stochastic processes with dynamic normalizing flows. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020; No. 654, pp. 7805–7815.
37. Guo, L.; Wu, H.; Zhou, T. Normalizing Field Flows: Solving forward and inverse stochastic differential equations using Physics-Informed flow model. *J. Comput. Phys.* **2022**, *461*, 11120244. [CrossRef]
38. Hodgkinson, L.; van der Heide, C.; Roosta, F.; Mahoney, M.W. Stochastic Normalizing Flows. *arXiv* **2020**, arXiv:2002.09547.
39. Papamakarios, G.; Nalisnick, E.; Rezende, D.J.; Mohamed, S.; Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.* **2021**, *22*, 1–64.
40. Urain, J.; Ginesi, M.; Tateo, D.; Peters, J. Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 25–29 October 2020.
41. Chen, Y.; Xiu, D.B. Learning stochastic dynamical systems via flow map operator. *J. Comput. Phys.* **2024**, *508*, 112984. [CrossRef]
42. Qin, T.; Wu, K.L.; Xiu, D.B. Data-driven governing equations approximation using deep neural networks. *J. Comput. Phys.* **2019**, *395*, 620–635. [CrossRef]
43. O'Leary, J.; Paulson, J.A.; Mesbah, A. Stochastic physics-informed neural ordinary differential equations. *J. Comput. Phys.* **2022**, *468*, 111466. [CrossRef]
44. Hairer, E.; Lubich, C.; Wanner, G. Symplectic Integration of Hamiltonian Systems. In *Geometric Numerical Integration*; Bank, R., Graham, R.L., Stoer, J., Varga, R., Yserentant, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 179–236.
45. Feng, K.; Wu, H.M.; Qin, M.Z.; Wang, D.L. Construction of canonical difference schemes for Hamiltonian formalism via generating functions. *J. Comp. Math.* **1989**, *7*, 71–96.
46. Deng, J.; Anton, C.A.; Wong, Y.S. High-order symplectic schemes for stochastic Hamiltonian systems. *Commun. Comput. Phys.* **2014**, *16*, 169–200. [CrossRef]
47. Hong, J.L.; Ruan, J.L.; Sun, L.Y.; Wang, L.J. Structure-preserving numerical methods for stochastic Poisson systems. *Commun. Comput. Phys.* **2021**, *29*, 802–830. [CrossRef]
48. Feng, L.; Gao, T.; Dai, M.; Duan, J. Auto-SDE: Learning effective reduced dynamics from data-driven stochastic dynamical systems. *arXiv* **2022**, arXiv:2205.04151.