NONSEPARABLE SYMPLECTIC NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Predicting the behaviors of Hamiltonian systems has been drawing increasing attention in scientific machine learning. However, the vast majority of the literature was focused on predicting separable Hamiltonian systems with their kinematic and potential energy terms being explicitly decoupled, while building data-driven paradigms to predict nonseparable Hamiltonian systems that are ubiquitous in fluid dynamics and quantum mechanics were rarely explored. The main computational challenge lies in the effective embedding of symplectic priors to describe the inherently coupled evolution of position and momentum, which typically exhibits intricate dynamics. To solve the problem, we propose a novel neural network architecture, Nonseparable Symplectic Neural Networks (NSSNNs), to uncover and embed the symplectic structure of a nonseparable Hamiltonian system from limited observation data. The enabling mechanics of our approach is an augmented symplectic time integrator to decouple the position and momentum energy terms and facilitate their evolution. We demonstrated the efficacy and versatility of our method by predicting a wide range of Hamiltonian systems, both separable and nonseparable, including chaotic vortical flows. We showed the unique computational merits of our approach to yield long-term, accurate, and robust predictions for large-scale Hamiltonian systems by rigorously enforcing symplectomorphism.

1 INTRODUCTION

A Hamiltonian dynamic system refers to a formalism for modeling a physical system exhibiting some specific form of energy conservation during its temporal evolution. A typical example is a pendulum whose total energy (referred as the system's Hamiltonian) is conserved as a temporally invariant sum of its kinematic energy and potential energy. Mathematically, such energy conservation indicates a specific geometric structure underpinning its time integration, named as a symplectic structure, which further spawns a wide range of numerical time integrators to model Hamiltonian systems. These symplectic time integrators have proven their effectiveness on simulating a variety of energy-conserving dynamics when Hamiltonian expressions are known as a prior. Examples encompass applications in plasma physics (Morrison, 2005), electromagnetics (Li et al., 2019), fluid mechanics (Salmon, 1988), and celestial mechanics (Saari & Xia, 1996), to name a few.

On another front, the emergence of the various machine learning paradigms with their particular focus on uncovering the hidden invariant quantities and their evolutionary structures enable a faithful prediction of Hamiltonian dynamics without knowing its analytical energy expression beforehand. The key mechanics underpinning these learning models lie in a proper embedding of the strong mathematical inductive priors to ensure Hamiltonian conservation in a neural network data flow. Typically, such priors are realized in a variational way or a structured way. For example, in Greydanus et al. (2019), the Hamiltonian conservation is encoded in the loss function. This category of methods does not assume any combinatorial pattern of the energy term and therefore relies on the inherent expressiveness of neural networks to distill the Hamiltonian structure from abundant training datasets (Choudhary et al., 2019). Another category of Hamiltonian networks, which we refer to as structured approaches, implements the conservation law indirectly by embedding a symplectic time integrator (DiPietro et al., 2020; Tong et al., 2020; Chen et al., 2020) or a composition of linear, activation and gradient modules (Jin et al., 2020) into the network architecture .

One of the main limitations of the current structured methods lies in the separable assumption of the Hamiltonian expression. Examples of separable Hamiltonian systems include the pendulum, the Lotka–Volterra (Zhu et al., 2016), the Kepler (Antohe & Gladwell, 2004), and the Hénon–Heiles

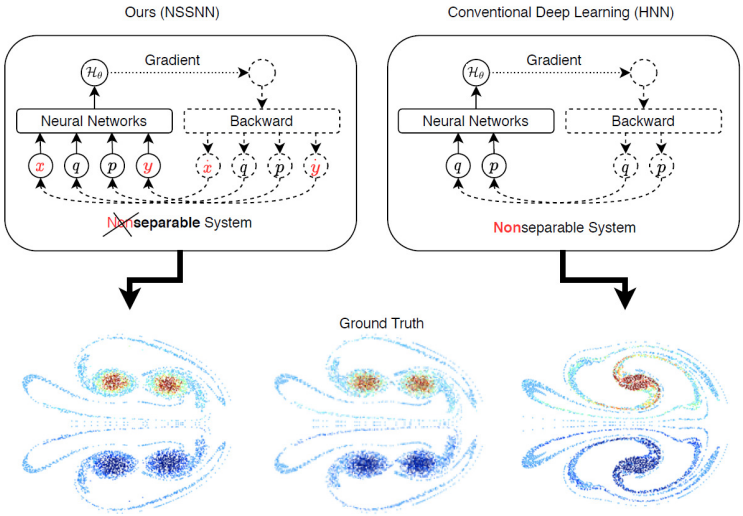

Figure 1: Comparison between NSSNN and HNN regarding the network design and prediction results of a vortex flow example.

.

systems (Zotos, 2015). However, beyond this scope, there exist various nonseparable systems whose Hamiltonian has no explicit expression to decouple the position and momentum energies. Examples include incompressible flows (Suzuki et al., 2007), quantum systems (Bonnabel et al., 2009), rigid body dynamics (Chadaj et al., 2017), charged particle dynamics (Zhang et al., 2016), and nonlinear Schrödinger equation (Brugnano et al., 2018). This nonseparability typically causes chaos and instability, which further complicates the systems' dynamics. Although SympNet in Jin et al. (2020) can be used to learn and predict nonseparable Hamiltonian systems, multiple matrices of the same order with system dimension are needed in the training process of SymNet, resulting difficulties in generalizing into high-dimensional large-scale N-body problems which are common in a series of nonseparable Hamiltonian systems, such as quantum multibody problems and vortex-particle dynamics problems. Such chaotic and large-scale nature jointly add shear difficulties for a conventional machine learning model to deliver faithful predictions.

In this paper, we propose an effective machine learning paradigm to predict nonseparable Hamiltonian systems. We build a novel neural network architecture, named as nonseparable symplectic neural networks (NSSNNs), to enable accurate and robust predictions of long-term Hamiltonian dynamics based on short-term observation data. Our proposed method belongs to the category of structured network architectures: it intrinsically embeds the symplectomorphism into the network design to strictly preserve the symplectic evolution and further conserves the unknown, nonseparable Hamiltonian energy. The enabling techniques we adopted in our learning framework consist in an augmented symplectic time integrator to asymptotically "decouple" the position and momentum quantities that were nonseparable in their original form. We also introduce Lagrangian multiplier in the augmented phase space to improve the system's numerical stability. Our network design is motivated by ideas originated from physics (Tao, 2016) and optimization (Boyd et al., 2004). The combination of these mathematical observations and numerical paradigms enables a novel neural network architecture that can drastically enhances both the scale and scope of the current predictions.

We show a motivational example in Figure 1 by comparing our approach with a traditional HNN method (Greydanus et al., 2019) regarding their structural designs and predicting abilities. We refer the readers to Section 6 for a detailed discussion. As shown in Figure 1, the vortices evolved using NSSNN are separated nicely as the ground truth, while the vortices merge together using HNN due to the failure of conserving the symplectic structure of a nonseparable system. The conservative capability of NSSNN springs from our design of the auxiliary variables (red x and y) which converts the original nonseparable system into a higher dimensional quasi-separable system where we can adopt a symplectic integrator.

2 RELATED WORKS

Data-driven physical prediction. Data-driven approaches have been widely applied in physical systems including fluid mechanics (Brunton et al., 2020), wave physics (Hughes et al., 2019), quantum physics (Sellier et al., 2019), thermodynamics (Hernandez et al., 2020), and material science (Teicherta et al., 2019). Among these different physical systems, data-driven fluid receives increasing attention. We refer the readers to Brunton et al. (2020) for a thorough survey of the fundamental machine learning methodologies as well as their uses for understanding, modeling, optimizing, and controlling fluid flows in experiments and simulations based on training data. One of the motivations of our work is to design a versatile learning approach that can predict complex fluid motions. On another front, many pieces of research focus on incorporating physical priors into the learning framework, e.g., by enforcing incompressibility (Mohan et al., 2020), the Galilean invariance (Ling et al., 2016), quasistatic equilibrium (Geng et al., 2020), the Lagrangian invariance (Cranmer et al., 2020), and Hamiltonian conservation (Hernandez et al., 2020; Greydanus et al., 2019; Jin et al., 2020; Zhong et al., 2020). Here, inspired by the idea of embedding physics priors into neural networks, we aim to accelerate the learning process and improve the accuracy of our model.

Neural networks for Hamiltonian systems. Greydanus et al. (2019) introduced Hamiltonian neural networks (HNNs) to conserve the Hamiltonian energy of the system by reformulating the loss function. Inspired by HNN, a series of methods intrinsically embedding a symplectic integrator into the recurrent neural network were proposed, such as SRNN (Chen et al., 2020), TaylorNet (Tong et al., 2020) and SSINN (DiPietro et al., 2020), to solve separable Hamiltonian systems. Combined with graph networks (Sanchez-Gonzalez et al., 2019; Battaglia et al., 2016), these methods were further generalized to large-scale N-body problems induced by interaction force between the particle pairs. Jin et al. (2020) proposed SympNet by directly constructing the symplectic mapping of system variables within neighbouring time steps to handle both separable and nonseparable Hamiltonian systems. However, the scale of parameters in SympNet for trainning N dimensional Hamiltonian system is $O(N^2)$, which makes it hard to be generalized to the high dimensional N-body problems. Our NSSNN overcomes these limitations by devising a new Hamiltonian network architecture that is specifically suited for nonseparable systems (see details in Section 5). In addition, the Hamiltonian based neural networks can be extended to further applications. Toth et al. (2020) developed the Hamiltonian Generative Network (HGN) to learn Hamiltonian dynamics from high-dimensional observations (such as images). Moreover, Zhong et al. (2020) introduced Symplectic ODE-Net (SymODEN), which adds an external control term to the standard Hamiltonian dynamics.

3 FRAMEWORK

3.1 AUGMENTED HAMILTONIAN EQUATION

We start by considering a Hamiltonian system with N pairs of canonical coordinates (i.e. N generalized positions and N generalized momentum). The time evolution of canonical coordinates is governed by the symplectic gradient of the Hamiltonian (Hand & Finch, 2008). Specifically, the time evolution of the system is governed by Hamilton's equations as

$$\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{q}}, \tag{1}$$

with the initial condition $(\boldsymbol{q}, \boldsymbol{p})|_{t=t_0} = (\boldsymbol{q}_0, \boldsymbol{p}_0)$. In a general setting, $\boldsymbol{q} = (q_1, q_2, \cdots, q_N)$ represents the positions and $\boldsymbol{p} = (p_1, p_2, ...p_N)$ denotes their momentum. Function $\mathcal{H} = \mathcal{H}(\boldsymbol{q}, \boldsymbol{p})$ is the Hamiltonian, which corresponds to the total energy of the system. An important feature of Hamilton's equations is its symplectomorphism (see Appendix B for a detailed overview).

The symplectic structure underpinning our proposed network architecture draws inspirations from the original research of Tao (2016) in computational physics. In Tao (2016), a generic, high-order, explicit and symplectic time integrator was proposed to solve (1) of an arbitrary separable and nonseparable Hamiltonian $\mathcal{H}$. This is implemented by considering an augmented Hamiltonian

$$\overline{\mathcal{H}}(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) := \mathcal{H}_A + \mathcal{H}_B + \omega \mathcal{H}_C \tag{2}$$

with

$$\mathcal{H}_A = \mathcal{H}(\boldsymbol{q}, \boldsymbol{y}), \;\; \mathcal{H}_B = \mathcal{H}(\boldsymbol{x}, \boldsymbol{p}), \;\; \mathcal{H}_C = \frac{1}{2}\left(\|\boldsymbol{q} - \boldsymbol{x}\|_2^2 + \|\boldsymbol{p} - \boldsymbol{y}\|_2^2\right) \tag{3}$$

in an extended phase space with symplectic two form $\mathrm{d}\boldsymbol{q} \wedge \mathrm{d}\boldsymbol{p} + \mathrm{d}\boldsymbol{x} \wedge \mathrm{d}\boldsymbol{y}$, where ω is a constant that controls the binding of the original system and the artificial restraint.

Notice that the Hamilton's equations for $\overline{\mathcal{H}}$

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \dfrac{\partial \overline{\mathcal{H}}}{\partial \boldsymbol{p}} = \dfrac{\partial \mathcal{H}(\boldsymbol{x}, \boldsymbol{p})}{\partial \boldsymbol{p}} + \omega(\boldsymbol{p} - \boldsymbol{y}), \\ \dfrac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\dfrac{\partial \overline{\mathcal{H}}}{\partial \boldsymbol{q}} = -\dfrac{\partial \mathcal{H}(\boldsymbol{q}, \boldsymbol{y})}{\partial \boldsymbol{q}} - \omega(\boldsymbol{q} - \boldsymbol{x}), \\ \dfrac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \dfrac{\partial \overline{\mathcal{H}}}{\partial \boldsymbol{y}} = \dfrac{\partial \mathcal{H}(\boldsymbol{q}, \boldsymbol{y})}{\partial \boldsymbol{y}} - \omega(\boldsymbol{p} - \boldsymbol{y}), \\ \dfrac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t} = -\dfrac{\partial \overline{\mathcal{H}}}{\partial \boldsymbol{x}} = -\dfrac{\partial \mathcal{H}(\boldsymbol{x}, \boldsymbol{p})}{\partial \boldsymbol{x}} + \omega(\boldsymbol{q} - \boldsymbol{x}), \end{cases} \tag{4}$$

with the initial condition $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})|_{t=t_0} = (\boldsymbol{q}_0, \boldsymbol{p}_0, \boldsymbol{q}_0, \boldsymbol{p}_0)$ have the same exact solution as (1) in the sense that $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{p})$. Hence, we can get the solution of (1) by solving (4). Furthermore, it is possible to construct high-order symplectic integrators for $\overline{\mathcal{H}}$ in (4) with explicit updates. Our model aims to learn the dynamical evolution of $(\boldsymbol{q}, \boldsymbol{p})$ in (1) by embedding (4) into the framework of NeuralODE (Chen et al., 2018). The coefficient ω acts as a regularizer, which stabilizes the numerical results (see Section 4).

3.2 Nonseparable Hamiltonian Neural Network

We learn the nonseparable Hamiltonian dynamics (1) by constructing an augmented system (4), from which we can obtain the energy function $\mathcal{H}(\boldsymbol{q}, \boldsymbol{p})$ by training the neural network $\mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{p})$ with parameter $\boldsymbol{\theta}$ and calculate the gradient $\boldsymbol{\nabla}\mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{p})$ by taking the in-graph gradient. For the constructed network $\mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{p})$, we integrate (4) by using the second-order symplectic integrator (Tao, 2016). Specifically, we will have an input layer $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{q}_0, \boldsymbol{p}_0, \boldsymbol{q}_0, \boldsymbol{p}_0)$ at $t = t_0$ and an output layer $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{q}_n, \boldsymbol{p}_n, \boldsymbol{x}_n, \boldsymbol{y}_n)$ at $t = t_0 + n\mathrm{d}t$.

Algorithm 1 Integrate (4) by using the second-order symplectic integrator

Input: $\boldsymbol{q}_0, \boldsymbol{p}_0, t_0, t, \mathrm{d}t$; $\boldsymbol{\phi}_1^\delta$, $\boldsymbol{\phi}_2^\delta$, and $\boldsymbol{\phi}_3^\delta$ in (5);
Output: $(\hat{\boldsymbol{q}}, \hat{\boldsymbol{p}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = (\boldsymbol{q}_n, \boldsymbol{p}_n, \boldsymbol{x}_n, \boldsymbol{y}_n)$

```
(q_0, p_0, x_0, y_0) = (q_0, p_0, q_0, p_0)
n = floor[(t − t_0)/dt]
for i = 1 → n do
    (q_i, p_i, x_i, y_i) = φ_1^{dt/2} ∘ φ_2^{dt/2} ∘ φ_3^{dt} ∘ φ_2^{dt/2} ∘
    φ_1^{dt/2} ∘ (q_{i−1}, p_{i−1}, x_{i−1}, y_{i−1});
end
```

The recursive relations of $(\boldsymbol{q}_i, \boldsymbol{p}_i, \boldsymbol{x}_i, \boldsymbol{y}_i)$, $i = 1, 2, \cdots, n$, can be expressed by the algorithm 1 (also see Figure 8 in Appendix A). The input functions $\boldsymbol{\phi}_1^\delta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{\phi}_2^\delta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$, and $\boldsymbol{\phi}_3^\delta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$ in algorithm 1 are

$$\begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{p} - \delta[\partial \mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{y})/\partial \boldsymbol{q}] \\ \boldsymbol{x} + \delta[\partial \mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{y})/\partial \boldsymbol{p}] \\ \boldsymbol{y} \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{q} + \delta[\partial \mathcal{H}_\theta(\boldsymbol{x}, \boldsymbol{p})/\partial \boldsymbol{p}] \\ \boldsymbol{p} \\ \boldsymbol{x} \\ \boldsymbol{y} - \delta[\partial \mathcal{H}_\theta(\boldsymbol{x}, \boldsymbol{p})/\partial \boldsymbol{q}] \end{bmatrix}, \text{ and } \frac{1}{2} \begin{bmatrix} \begin{pmatrix} \boldsymbol{q} + \boldsymbol{x} \\ \boldsymbol{p} + \boldsymbol{y} \end{pmatrix} + \boldsymbol{R}^\delta \begin{pmatrix} \boldsymbol{q} - \boldsymbol{x} \\ \boldsymbol{p} - \boldsymbol{y} \end{pmatrix} \\ \begin{pmatrix} \boldsymbol{q} + \boldsymbol{x} \\ \boldsymbol{p} + \boldsymbol{y} \end{pmatrix} - \boldsymbol{R}^\delta \begin{pmatrix} \boldsymbol{q} - \boldsymbol{x} \\ \boldsymbol{p} - \boldsymbol{y} \end{pmatrix} \end{bmatrix}, \tag{5}$$

respectively. Here

$$\boldsymbol{R}^\delta := \begin{bmatrix} \cos(2\omega\delta)\boldsymbol{I} & \sin(2\omega\delta)\boldsymbol{I} \\ -\sin(2\omega\delta)\boldsymbol{I} & \cos(2\omega\delta)\boldsymbol{I} \end{bmatrix}, \text{ where } \boldsymbol{I} \text{ is a identity matrix.} \tag{6}$$

We remark that $\boldsymbol{x}$ and $\boldsymbol{y}$ are just auxiliary variables, which are theoretically equal to $\boldsymbol{q}$ and $\boldsymbol{p}$. Therefore, we can use the data set of $(\boldsymbol{q}, \boldsymbol{p})$ to construct the data set containing variables $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$. In addition, by constructing the network $\mathcal{H}_\theta$, we show that theorem B.1 in Appendix B holds, so the networks $\boldsymbol{\phi}_1^\delta$, $\boldsymbol{\phi}_2^\delta$, and $\boldsymbol{\phi}_3^\delta$ in (5) preserve the symplectic structure of the system. Suppose that Φ_1 and Φ_2 are two symplectomorphisms. Then, it is easy to show that their composite map $\Phi_2 \circ \Phi_1$ is also symplectomorphism due to the chain rule. Thus, the symplectomorphism of algorithm 1 can be guaranteed by the theorems B.1.

4 Training settings and ablation tests

We use 6 linear layers with hidden size 64 to model $\mathcal{H}_\theta$, all of which are followed by a Sigmoid activation function except the last one. The derivatives $\partial \mathcal{H}_\theta/\partial \boldsymbol{p}, \partial \mathcal{H}_\theta/\partial \boldsymbol{q}, \partial \mathcal{H}_\theta/\partial \boldsymbol{x}, \partial \mathcal{H}_\theta/\partial \boldsymbol{y}$ are

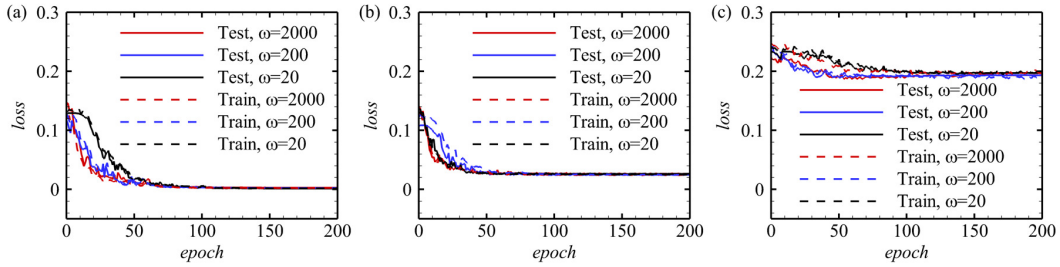

Figure 2: Comparisons of training and validation losses with different ω in the training process. The solid and dashed lines represent training and validation losses respectively. The networks are trained by (a) the clean dataset, (b) the dataset with noise $\sim 0.05U(-1,1)$, and (c) the dataset with noise $\sim 0.4U(-1,1)$.

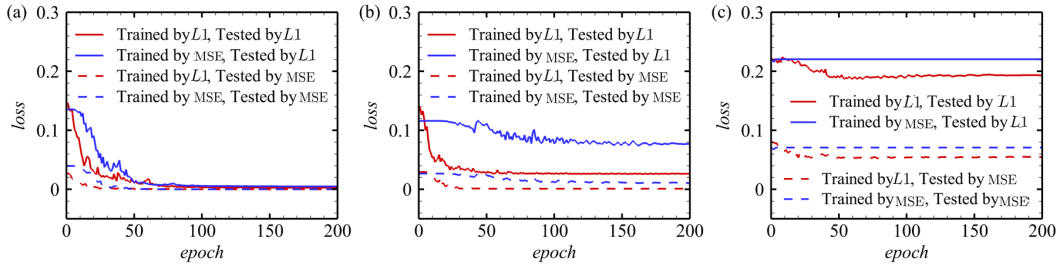

Figure 3: Comparisons of validation losses with different training loss functions in the training process. The red solid and dashed lines represent the networks trained by $L1$ loss function and validated by $L1$ and MSE respectively; the blue solid and dashed lines represent the networks trained by MSE and validated by $L1$ and MSE respectively. The networks are trained by (a) the clean dataset, (b) the dataset with noise $\sim 0.05U(-1,1)$, and (c) the dataset with noise $\sim 0.4U(-1,1)$.

all obtained by automatic differentiation in Pytorch (Paszke et al., 2019). The weights of the linear layers are initialized by Xavier initializaiton (Glorot & Bengio, 2010).

We generate the dataset for training and validation using high-precision numerical solver (Tao, 2016), where the ratio of training and validation datasets is $9:1$. We set the dataset $(\boldsymbol{q}_0^j, \boldsymbol{p}_0^j)$ as the start input and $(\boldsymbol{q}^j, \boldsymbol{p}^j)$ as the target with $j = 1, 2, \cdots, N_s$, and the time span between $(\boldsymbol{q}_0^j, \boldsymbol{p}_0^j)$ and $(\boldsymbol{q}^j, \boldsymbol{p}^j)$ is T_{train}. Feeding $(\boldsymbol{q}_0, \boldsymbol{p}_0) = (\boldsymbol{q}_0^j, \boldsymbol{p}_0^j),\ t_0 = 0,\ t = T_{train}$, and time step d$t$ in Algorithm 1 to get the predicted variables $(\hat{\boldsymbol{q}}^j, \hat{\boldsymbol{p}}^j, \hat{\boldsymbol{x}}^j, \hat{\boldsymbol{y}}^j)$. Accordingly, the loss function is defined as

$$\mathcal{L}_{NSSNN} = \frac{1}{N_b}\sum_{j=1}^{N_b} \|\boldsymbol{q}^{(j)} - \hat{\boldsymbol{q}}^{(j)}\|_1 + \|\boldsymbol{p}^{(j)} - \hat{\boldsymbol{p}}^{(j)}\|_1 + \|\boldsymbol{q}^{(j)} - \hat{\boldsymbol{x}}^{(j)}\|_1 + \|\boldsymbol{p}^{(j)} - \hat{\boldsymbol{y}}^{(j)}\|_1, \quad (7)$$

where $N_b = 512$ is the batch size of the training samples. We use the Adam optimizer (Kingma & Ba, 2015) with learning rate 0.05. The learning rate is multiplied by 0.8 for every 10 epoches.

Taking system $\mathcal{H}(q,p) = 0.5(q^2+1)(p^2+1)$ as an example, we carry out a series of ablation tests based on our constructed networks. Normally, we set the time span, time step and dateset size as $T = 0.01$, d$t = 0.01$ and $N_s = 1280$.

The choice of ω in (4) is largely flexible, since NSSNN is not sensitive to the parameter ω when it is larger than a certain threshold. Figure 2 shows the training and validation losses with different ω in the network trained by clean and noise datasets. Though the convergence rates are slightly different in a small scope, the examples with various ω are able to converge to the same size of training and validation losses. Here, we set $\omega = 2000$, but ω can be smaller than 2000. The only requirement for picking ω is that it has to be larger than $O(10)$, which is detailed in Appendix C.

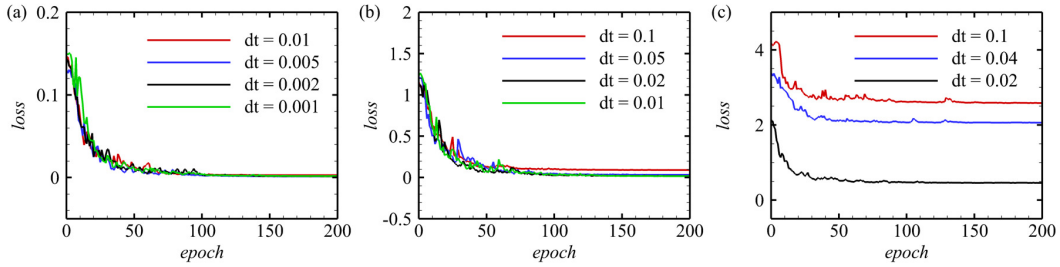

Figure 4: Comparisons of validation losses with different dt in the training process. (a), (b), and (c) are trained based on different time spans $T_{train} = 0.01$ 0.1, and 0.2, respectively.

We pick $L1$ loss function to train our network due to its better performance. Figure 3 compares the validation losses with different training loss functions in the network trained by clean and noise datasets. Figure 3(a) shows that either the network trained by $L1$ or MSE with clean dataset can converge to a small validation loss, but the network trained by $L1$ loss converges relatively faster. Figures 3(b) and 3(c) both show that the network trained by $L1$ with noise dataset can converge to a smaller validation loss. In addition, we already introduced a regularization term in the symplectic integrator embedded in the network; thus, there is no need to add the regularization term in the loss function.

The integral time step in the sympletic integrator is a vital parameter, and the choice of dt largely depends on the time span T_{train}. Figure 4 compares the validation losses generated by various integral time steps dt based on fixed dataset time spans $T_{train} = 0.01$, 0.1 and 0.2 respectively in the training process. The validation loss converges to a similar degree with various dt based on fixed $T_{train} = 0.01$ and $T_{train} = 0.1$ in 4(a) and (b), while it increases significantly as dt increases based on fixed $T_{train} = 0.02$ in 4(c). Thus, we should take relatively small dt for the dataset with larger time span T_{train}

5 Comparisons with other methods

5.1 Methodologies

We compare our method with other recently proposed methods, such as HNN (Greydanus et al., 2019), NeuralODE (Chen et al., 2018), TaylorNet (Tong et al., 2020), SSINN (DiPietro et al., 2020), SRNN (Chen et al., 2020), and SympNet (Jin et al., 2020). There are several features distinguishing our method with others, as shown in Table 1. HNN first enforces conservative features of a Hamiltonian system by reformulating its loss function, which incurs two main shortcomings. On the one hand, it requires the temporal derivatives of the momentum and the position of the systems to calculate the loss function, which is difficult to obtain from real-world systems. On the other hand, HNN doesn't strictly preserve the symplectic structure, because its symplectomorphism is realized by its loss function rather than its intrinsic network architecture. NeuralODE successfully bypasses the time derivatives of the datasets by incorporating an integrator solver into the network architecture.

Embedding the Hamiltonian prior into the NeuralODE, a series of methods are proposed, such as SRNN, SSINN, and TaylorNet, to predict the continuous trajectory of system variables; however, presently these methods are only designed to solve separable Hamiltonian systems. Instead of updating the continuous dynamics by integrating the neural networks in NeuralODE, SympNet adopts a symplectomorphism composed of well-designed both linear and non-linear matrices to intrinsically map the system variables within neighbouring time steps. However, the parameters scale in the matrix map for trainning N dimensional Hamiltonian system in SympNet is $O(N^2)$, which makes it hard to generalize to the high dimensional N-body problems. For example, in Section 6, we predict the dynamic evolution of 6000 vortex particles, which is challenging for the training process of the SympNet on the level of $O(6000^2)$.

NSSNN overcomes the weaknesses mentioned above. Under the framework of NeuralODE, NSSNN utilizes continuously-defined dynamics in the neural networks, which gives it the capability to learn

Table 1: Comparison between HNN (Greydanus et al., 2019), NeuralODE (Chen et al., 2018), TaylorNet (Tong et al., 2020), SSINN (DiPietro et al., 2020), SRNN (Chen et al., 2020), SympNet (Jin et al., 2020), and NSSNN. ✓ represents the method preserves such property.

Methods	NSSNN	HNN	NeuralODE	TaylorNet	SSINN	SRNN	SympNet
Solve nonseparable systems	✓	✓	✓				✓
Solve separable systems	✓	✓	✓	✓	✓	✓	✓
Preserve symplectic structure	✓	Partially		✓	Partially	✓	✓
Utilize continuous dynamics	✓		✓	✓	✓	✓	✓
No need for derivatives in dataset	✓		✓	✓	✓	✓	✓
Long-term predictability	✓	Partially		✓	✓	✓	✓
Extend to N-body system	✓	✓	✓	✓		✓	

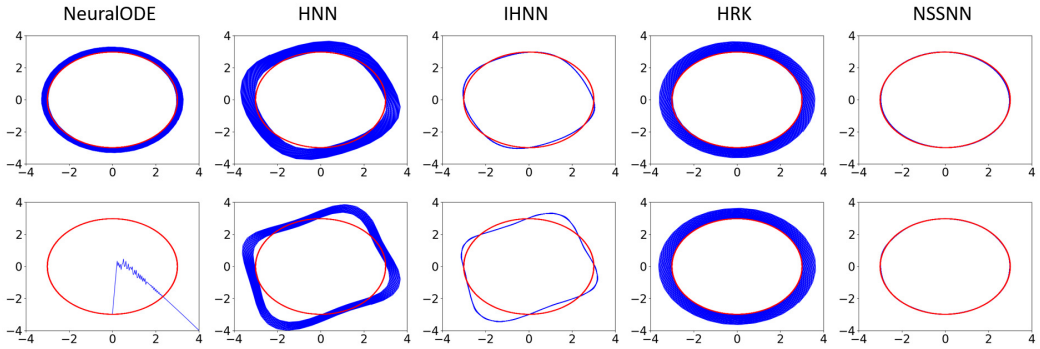

Figure 5: Comparison of prediction results of (q, p) for the spring system $\mathcal{H} = 0.5(q^2 + p^2)$ from $t = 0$ to $t = 200$ with $(q_0, p_0) = (0, -3)$. The time span of the datasets are $T_{train} = 0.4$ (first row) and $T_{train} = 1$ (second row). The five columns are five different methods NeuralODE, HNN, IHNN, HRK, and NSSNN, respectively. The red line denotes the ground truth; the blue line denotes the prediction, which are perfectly overlapping in NSSNN. The prediction ability of HNN and IHNN improves significantly with the decreasing of T_{train} of the dataset which however may be hard to obtain in the actual experimental measurements.

the continuous time evolution of dynamical systems. Based on Tao (2016), NSSNN embeds the symplectic proir into the nonseparable symplectic integrator to ensure the strict symplectomorphism, thereby guaranteeing the property of long-term predictability. In addition, unlike SympNet, NSSNN is highly flexible and can be generalized to high dimensional N-body problems by involving the interaction networks (Sanchez-Gonzalez et al., 2019), which will be further discussed in Section 6.

5.2 Experiments

We compare five implementations that learn and predict Hamiltonian systems. The first one is NeuralODE, which trains the system by embedding the network $\boldsymbol{f}_\theta \to (\mathrm{d}\boldsymbol{q}/\mathrm{d}t, \mathrm{d}\boldsymbol{p}/\mathrm{d}t)$ into the Runge-Kutta (RK) intergrator. The other four, however, achieve the goal by fitting the Hamiltonian $\mathcal{H}_\theta \to \mathcal{H}$ based on (1). Specifically, HNN trains the network with the constraints of Hamiltonian symplectic gradient along with the time derivative of system variables and then embeds the well-trained $\mathcal{H}_\theta$ into the RK integrator for predicting the system. The third and fourth implementations are ablation tests. One of them is improved HNN (IHNN), which embeds the well-trained $\mathcal{H}_\theta$ into the nonseparable symplectic integrator (Tao's integrator) for predicting. The other is to directly embed $\mathcal{H}_\theta$ into the RK integrator for training, which we call it HRK. The fifth method is NSSNN, which embeds $\mathcal{H}_\theta$ into the nonseparable symplectic integrator for training.

For fair comparison, we adopt the same network structure (except that the dimension of output layer in NeuralODE is two times larger than that in the other four), the same $L1$ loss function and same size of the dataset, and the precision of all integral schemes is second order, and the other parameters keep consistent with the one in Section 4. The time derivative in the dataset for training HNN and

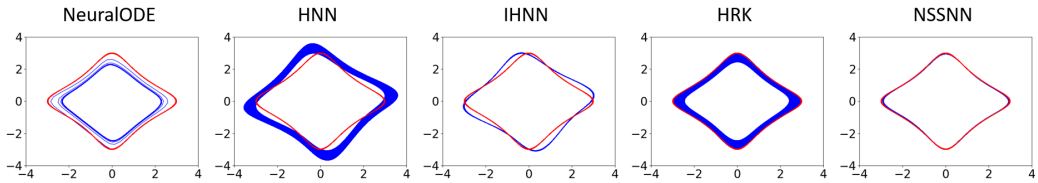

Figure 6: Comparison of prediction results of (q, p) for the Tao's system $\mathcal{H} = 0.5(q^2 + 1)(p^2 + 1)$ from $t = 0$ to $t = 20000$ with $(q_0, p_0) = (0, -3)$. The network is trained by the dataset with noise $\sim 0.05U(-1, 1)$, and the time span of the dataset is $T_{train} = 0.2$. The red line denotes the ground truth; the blue line denotes the prediction, which are perfectly overlapping in NSSNN.

IHNN is obtained by the first difference method

$$\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} \approx \frac{\boldsymbol{q}(T_{train}) - \boldsymbol{q}(0)}{T_{train}} \quad \text{and} \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} \approx \frac{\boldsymbol{p}(T_{train}) - \boldsymbol{q}(0)}{T_{train}}. \tag{8}$$

Figure 5 demonstrates the differences between the five methods using a spring system $\mathcal{H} = 0.5(q^2 + p^2)$ with different time span $T_{train} = 0.4,\ 1$ and same time step $\mathrm{d}t = 0.2$. We can see that by introducing the nonseparable symplectic integrator into the prediction of Hamiltonian system, NSSNN has a stronger long-term predicting ability than all the other methods. In addition, the prediction of HNN and IHNN lies in the dataset with time derivative; consequently, it will lead to a larger error when the given time span T_{train} is large.

Moreover, the datasets obtained by (11) in HNN and IHNN are sensitive to noise. Figure 6 compares the predictions of (q, p) for the system $\mathcal{H} = 0.5(q^2 + 1)(p^2 + 1)$, where the network is trained by the dataset with noise $\sim 0.05U(-1, 1)$, along with time span $T_{train} = 0.2$ and time step $\mathrm{d}t = 0.02$. Under the condition with noise, NSSNN still performs well compared with other methods. Also, we compare the convergent error of a series of Hamiltonian systems with different $\mathcal{H}$ trained with noisy data in Appendix D, which generally shows better robustness than HNN does.

6 Modeling vortex dynamics of multi-particle system

For two-dimensional vortex particle systems, the dynamical equations of particle positions $(x_j, y_j),\ j = 1, 2, \cdots, N_v$ with particle strengths Γ_j can be written in the generalized Hamiltonian form as

$$\Gamma_j \frac{\mathrm{d}x_j}{\mathrm{d}t} = -\frac{\partial \mathcal{H}^p}{\partial y_j}, \quad \Gamma_j \frac{\mathrm{d}y_j}{\mathrm{d}t} = \frac{\partial \mathcal{H}^p}{\partial x_j}, \quad \text{with} \quad \mathcal{H}^p = \frac{1}{4\pi} \sum_{j,k=1}^{N_v} \Gamma_j \Gamma_k \log(|x_j - x_k|). \tag{9}$$

By including the given particle strengths Γ_j in Algorithm 1, we can still adopt the method mentioned above to learn the Hamiltonian in (9) when there are less particles. However, considering a system with $N_v \gg 2$ particles, the cost to collect training data from all N_v particles might be high, and the training process can be time-consuming. Thus, instead of collecting information from all N_v particles to train our model, we only use data collected from two bodies as training data to make prediction of the dynamics of N_v particles.

Specifically, we assume the interactive models between particle pairs with unit particle strengths $\Gamma_j = 1$ are the same, and their corresponding Hamiltonian can be represented as network $\hat{\mathcal{H}}_\theta(\boldsymbol{x}_j, \boldsymbol{x}_k)$, based on which the corresponding Hamiltonian of N_v particles can be written as (Battaglia et al., 2016; Sanchez-Gonzalez et al., 2019)

$$\mathcal{H}_\theta^p = \sum_{i,j=1}^{N_v} \Gamma_j \Gamma_k \hat{\mathcal{H}}_\theta(\boldsymbol{x}_j, \boldsymbol{x}_k). \tag{10}$$

We embed (10) into the symplectic integrator that includes Γ_j to obtain the final network architecture.

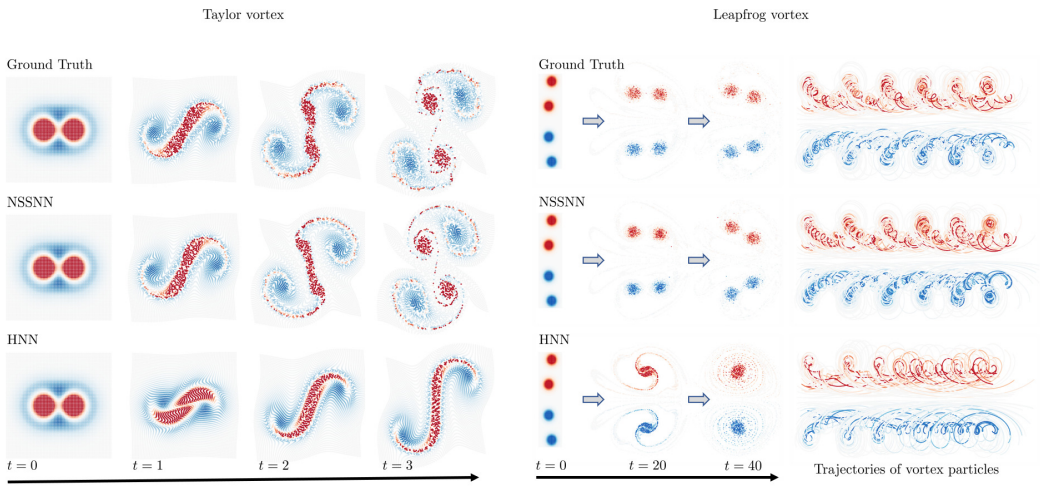

Figure 7: Taylor and Leapfrog vortex. We generate results of Taylor vortex and Leapfrop vortex using NSSNN and HNN, and compare them with the ground truth. 6000 vortex elements are used with corresponding initial vorticity conditions of Taylor vortex and Leapfrop vortex.

The setup of the multi-particle problem is similar to the previous problems. The training time span is $T_{train} = 0.01$ while the prediction period can be up to $T_{predict} = 40$. We use 2048 clean data samples to train our model. The training process takes about 100 epochs for the loss to converge. In Figure 7, we use our trained model to predict the dynamics of 6000-particle systems, including Taylor and Leapfrog vortices. We generate results of Taylor vortex and Leapfrop vortex using NSSNN and HNN and compare them with the ground truth. Vortex elements are used with corresponding initial vorticity conditions of Taylor vortex and Leapfrop vortex (Qu et al., 2019). The difficulty of the numerical modeling of these two systems lies in the separation of different dynamical vortices instead of having them merging into a bigger structure. In both cases, the vortices evolved using NSSNN are separated nicely as the ground truth shows, while the vortices merge together using HNN.

7 Limitations

The network with the embeded integrator is often more time-consuming to train than the one based on the dataset with time derivative. For example, the ratio of training time of the methods HNN and NSSNN is $1:3$ when $\mathrm{d}t = T_{train}$, and the training time of the recurrent networks further increases with the decreasing of $\mathrm{d}t$. Although a smaller $\mathrm{d}t$ often has higher discretization accuracy, there is a tradeoff between training cost and predicating accuracy. Additionally, a smaller $\mathrm{d}t$ may potentially cause gradient explosion. In this case, we may want to use adjoint method instead. Another limitation lies on the assumption that the symplectic structure is conserved. In real-world systems, there could be dissipation that makes this assumption unsatisfied.

8 Conclusions

We incorporate a classic ideal that maps a nonseparable system to a higher dimensional space making it quasi-separable to construct sympletic networks. With the intrinsic symplectic structure, NSSNN possesses many benefits compared with other methods. In particular, NSSNN is the first method that can learn the vortex dynamical system, and accurately predict the evolution of complex vortex structures, such as Taylor and Leapfrog vortices. NSSNN, based on the first principle learning complex systems, has potential applications in fields of physics, astronomy, and weather forecast, etc. We will further explore the possibilities of neural networks with inherent structure preserving ability in fields like 3D vortex dynamics and quantum turbulence. In addition, we will also work on general applications of NSSNN with datasets based on images or other real scenes through automatically identifying coordinate variables of Hamiltonian systems based on neural networks.

REFERENCES

V. Antohe and I. Gladwell. Performance of variable step size methods for solving model separable hamiltonian systems. *Math. COMPUT. MODEL.*, 40(11-12):1245–1262, 2004.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

S. Bonnabel, M. Mirrahimi, and P. Rouchon. Observer-based hamiltonian identification for quantum systems. *Automatica*, 45(5):1144–1155, 2009.

S. Boyd, S. P Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

L. Brugnano, C. Zhang, and D. Li. A class of energy-conserving hamiltonian boundary value methods for nonlinear schrödinger equation with wave operator. *Communications in Nonlinear Science and Numerical Simulation*, 60:33–49, 2018.

S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine Learning for Fluid Mechanics. *Annu. Rev. Fluid Mech.*, 52:477–508, 2020.

K. Chadaj, P. Malczyk, and J. Frączek. A parallel hamiltonian formulation for forward dynamics of closed-loop multibody systems. *Multibody Syst. Dyn.*, 39(1-2):51–77, 2017.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *Conference on Neural Information Processing Systems*, pp. 6571–6583, 2018.

Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic recurrent neural networks. In *International Conference on Learning Representations*, 2020.

Anshul Choudhary, John F Lindner, Elliott G Holliday, Scott T Miller, Sudeshna Sinha, and William L Ditto. Physics enhanced neural networks predict order and chaos. *arXiv preprint arXiv:1912.01958*, 2019.

M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. *arXiv:2003.04630*, 2020.

Daniel DiPietro, Shiying Xiong, and Bo Zhu. Sparse symplectically integrated neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

Z. Geng, D. Johnson, and R. Fedkiw. Coercing machine learning to output physically accurate results. *J. Comput. Phys.*, 406:109099, 2020.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *Conference on Neural Information Processing Systems*, pp. 15379–15389, 2019.

L. N. Hand and J. D. Finch. *Analytical Mechanics*. Cambridge University Press, 2008.

Q. Hernandez, A. Badias, D. Gonzalez, F. Chinesta, and E. Cueto. Structure-preserving neural networks. *arXiv:2004.04653*, 2020.

T. W. Hughes, I. A. D. Williamson, M. Minkov, and S. Fan. Wave physics as an analog recurrent neural network. *Sci. Adv.*, 5:6946, 2019.

P. Jin, A. Zhu, G. E. Karniadakis, and Y. Tang. Symplectic networks: intrinsic structure-preserving networks for identifying Hamiltonian systems. *arXiv:2001.03750*, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

A. N. Kolmogorov. On the Conservation of Conditionally Periodic Motions under Small Perturbation of the Hamiltonian. *Dokl. akad. nauk SSSR*, 98:527–530, 1954.

John Lee. *Introduction to topological manifolds*, volume 202. Springer Science & Business Media, 2010.

Y. Li, Y. He, J. Niesen Y. Sun, H. Qin, and J. Liu. Solving the vlasov–maxwell equations using hamiltonian splitting. *J. Comput. Phys.*, 396:381–399, 2019.

J. Ling, R. Jones, and J. Templeton. Machine learning strategies for systems with invariance properties. *J. Comput. Phys.*, 318:22–35, 2016.

A. T. Mohan, N. Lubbers, D. Livescu, and M. Chertkov. Embedding hard physical constraints in convolutional neural networks for 3d turbulence. In *International Conference on Learning Representations*, 2020.

P. J. Morrison. Hamiltonian and action principle formulations of plasma physics. *Phys. Plasmas*, 12: 058102, 2005.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

Z. Qu, X. Zhang, M. Gao, C. Jiang, and B. Chen. Efficient and conservative fluids using bidirectional mapping. *ACM Trans. Graph.*, 38:1–12, 2019.

D. G. Saari and Z. Xia. *Hamiltonian Dynamics and Celestial Mechanics*. American Mathematical Society, 1 edition, 1996.

R. Salmon. Hamiltonian fluid mechanics. *Ann. Rev. Fluid Mech.*, 20:225–256, 1988.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

J. M. Sellier, G. M. Caron, and J. Leygonie. Signed particles and neural networks, towards efficient simulations of quantum systems. *J. Comput. Phys.*, 387:154–162, 2019.

Y. Suzuki, S. Koshizuka, and Y. Oka. Hamiltonian moving-particle semi-implicit (hmps) method for incompressible fluid flows. *Computer Methods in Applied Mechanics and Engineering*, 196 (29-30):2876–2894, 2007.

M. Tao. Explicit symplectic approximation of nonseparable Hamiltonians: algorithm and long time performance. *Phys. Rev. E*, 94:043303, 2016.

G. H. Teicherta, A. R. Natarajanc, A. Van der Venc, and K. Garikipati. Machine learning materials physics: Integrable deep neural networks enable scale bridging by learning free energy functions. *Comput. Methods Appl. Mech. Engrg.*, 353:201–216, 2019.

Y. Tong, S. Xiong, X. He, G. Pan, and Bo Zhu. Symplectic neural networks in taylor series form for hamiltonian systems. *arXiv:2005.04986*, 2020.

P. Toth, D. J. Rezende, A. Jaegle, S. Racaniére, A. Botev, and I. Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations*, 2020.

R. Zhang, H. Qin, Y. Tang, J. Liu, Y. He, and J. Xiao. Explicit symplectic algorithms based on generating functions for charged particle dynamics. *Physical Review E*, 94(1):013205, 2016.

Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ode-net: learning hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020.

B. Zhu, R. Zhang, Y. Tang, X. Tu, and Y. Zhao. Splitting k-symplectic methods for non-canonical separable hamiltonian problems. *J. Comput. Phys.*, 322:387–399, 2016.

E. E Zotos. Classifying orbits in the classical hénon–heiles hamiltonian system. *Nonlinear Dynamics*, 79(3):1665–1677, 2015.

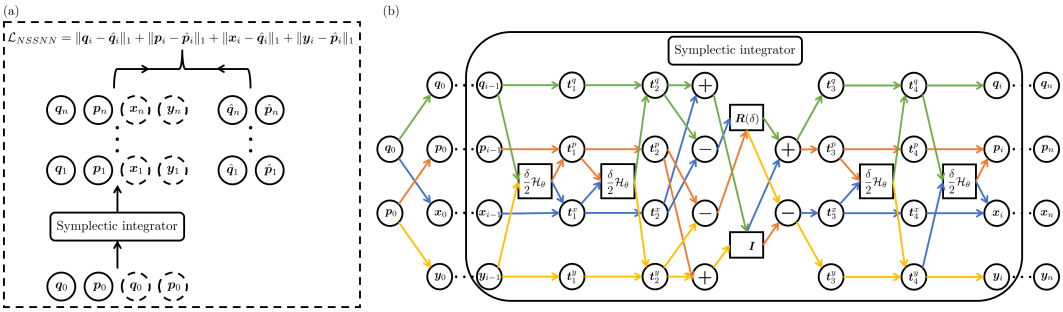

Figure 8: (a) The forward pass of an NSSNN is composed of a forward pass through a differentiable symplectic integrator as well as a backpropagation step through the model. (b) The schematic diagram of NSSNN.

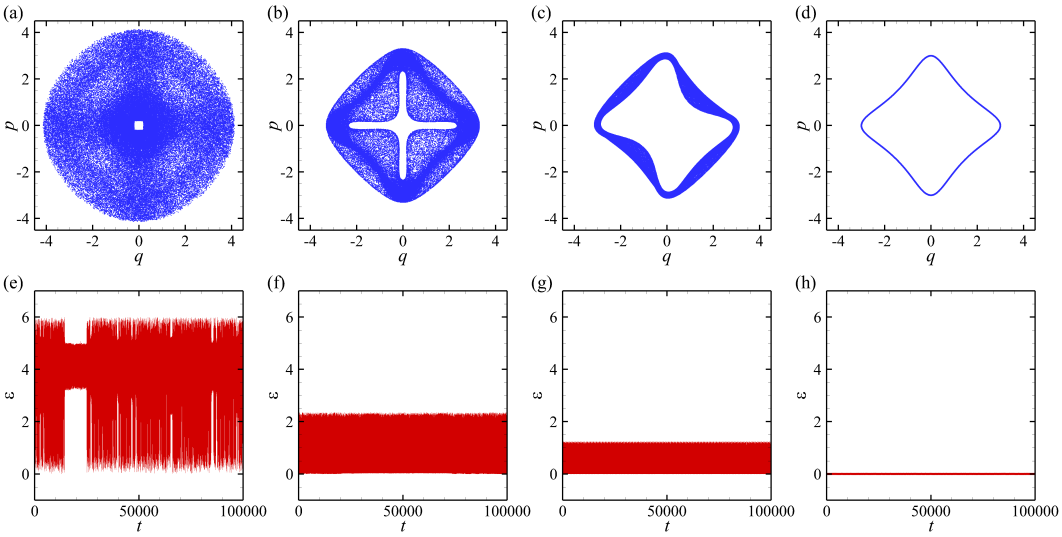

Figure 9: Comparison of different ω (from left to right columns) with (a) and (e) $\omega = 0$, (b) and (f) $\omega = 0.8$, (c) and (g) $\omega = 0.9$, (d) and (h) $\omega = 10$ in the symplectic integrator for nonseparable Hamiltonian $H = (q^2 + 1)(p^2 + 1)/2$. The upper row: projection of a trace $[q(t), p(t), x(t), y(t)]$ with $[q(0), p(0), x(0), y(0)] = (-3, 0, -3, 0)$ onto the $q - p$ plane. The bottom row: deviation $\epsilon = \|(q, p) - (x, y)\|_2$ between (q, p) and (x, y).

.

A Network Architecture

Figure 8(a) shows the forward pass of NSSNN is composed of a forward pass through a differentiable symplectic integrator as well as a backpropagation step through the model. Figure 8(b) plots the schematic diagram of NSSNN. For the constructed network $\mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{p})$, we integrate (4) by using the second-order symplectic integrator (Tao, 2016). Specifically, The input layer of the integrator is $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{q}_0, \boldsymbol{p}_0, \boldsymbol{q}_0, \boldsymbol{p}_0)$ at $t = t_0$ and the output layer is $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{q}_n, \boldsymbol{p}_n, \boldsymbol{x}_n, \boldsymbol{y}_n)$ at $t = t_0 + n\mathrm{d}t$. The recursive relations of $(\boldsymbol{q}_i, \boldsymbol{p}_i, \boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, 2, \cdots, n$, are expressed by the algorithm 1.

B Symplectomorphisms

One of the most important features of the time evolution of Hamilton's equations is that it is a symplectomorphism, representing a transformation of phase space that is volume-preserving. In the setting of canonical coordinates, symplectomorphism means the transformation of the phase flow of

a Hamiltonian system conserves the symplectic two-form

$$\mathrm{d}\boldsymbol{q} \wedge \mathrm{d}\boldsymbol{p} \equiv \sum_{j=1}^{N} \left(\mathrm{d}q_j \wedge \mathrm{d}p_j\right), \tag{11}$$

where $\wedge$ denotes the wedge product of two differential forms. The rules of wedge product can be found in Lee (2010). In the two dimensional case, (11) can be understood as the area element of surface. In this case, the symplectomorphism can be interpreted as the area element of the surface is constant. As proved below, our constructed network structure intrinsically preserves Hamiltonian structure.

Theorem B.1. *For a given δ, the mapping $\boldsymbol{\phi}_1^\delta$, $\boldsymbol{\phi}_2^\delta$, and $\boldsymbol{\phi}_3^\delta$ in (5) are symplectomorphisms.*

Proof. Let

$$(\boldsymbol{t}_j^q, \boldsymbol{t}_j^p, \boldsymbol{t}_j^x, \boldsymbol{t}_j^y) = \boldsymbol{\phi}_j^\delta(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y}), \quad j = 1, 2, 3. \tag{12}$$

From the first equation of (5), we have

$$\begin{aligned}
&\mathrm{d}\boldsymbol{t}_1^q \wedge \mathrm{d}\boldsymbol{t}_1^p + \mathrm{d}\boldsymbol{t}_1^x \wedge \mathrm{d}\boldsymbol{t}_1^y \\
=&\mathrm{d}\boldsymbol{q} \wedge \mathrm{d}\left[\boldsymbol{p} - \delta \frac{\partial \mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{y})}{\partial \boldsymbol{q}}\right] + \mathrm{d}\left[\boldsymbol{x} + \delta \frac{\partial \mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{y})}{\partial \boldsymbol{p}}\right] \wedge \mathrm{d}\boldsymbol{y} \\
=&\mathrm{d}\boldsymbol{q} \wedge \mathrm{d}\boldsymbol{p} + \mathrm{d}\boldsymbol{x} \wedge \mathrm{d}\boldsymbol{y} + \delta \left[\frac{\partial \mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{y})}{\partial \boldsymbol{q} \partial \boldsymbol{y}} - \frac{\partial \mathcal{H}_\theta(\boldsymbol{q}, \boldsymbol{y})}{\partial \boldsymbol{y} \partial \boldsymbol{q}}\right] \mathrm{d}\boldsymbol{q} \wedge \mathrm{d}\boldsymbol{y} \\
=&\mathrm{d}\boldsymbol{q} \wedge \mathrm{d}\boldsymbol{p} + \mathrm{d}\boldsymbol{x} \wedge \mathrm{d}\boldsymbol{y}.
\end{aligned} \tag{13}$$

Similarly, we can prove that $\mathrm{d}\boldsymbol{t}_2^q \wedge \mathrm{d}\boldsymbol{t}_2^p + \mathrm{d}\boldsymbol{t}_2^x \wedge \mathrm{d}\boldsymbol{t}_2^y = \mathrm{d}\boldsymbol{q} \wedge \mathrm{d}\boldsymbol{p} + \mathrm{d}\boldsymbol{x} \wedge \mathrm{d}\boldsymbol{y}$. In addition, from the third equation of (5), we can directly deduce that $\mathrm{d}\boldsymbol{t}_3^q \wedge \mathrm{d}\boldsymbol{t}_3^p + \mathrm{d}\boldsymbol{t}_3^x \wedge \mathrm{d}\boldsymbol{t}_3^y = \mathrm{d}\boldsymbol{q} \wedge \mathrm{d}\boldsymbol{p} + \mathrm{d}\boldsymbol{x} \wedge \mathrm{d}\boldsymbol{y}$. □

Suppose that Φ_1 and Φ_2 are two symplectomorphisms. Then, it is easy to show that their composite map $\Phi_2 \circ \Phi_1$ is also symplectomorphism due to the chain rule. Thus, the symplectomorphism of algorithm 1 can be guaranteed by the theorem B.1.

C Determining Coefficient ω

To further elucidation, the Hamiltonian $\mathcal{H}_A + \mathcal{H}_B$ without the binding, i.e., $\overline{\mathcal{H}}$ with $\omega = 0$, in extended phase space $(\boldsymbol{q}, \boldsymbol{p}, \boldsymbol{x}, \boldsymbol{y})$ may not be integrable, even if $\mathcal{H}(\boldsymbol{q}, \boldsymbol{p})$ is integrable in the original phase space $(\boldsymbol{q}, \boldsymbol{p})$. However, $\mathcal{H}_C$ is integrable. Thus, as ω increases, a larger proportion in the phase space for $\overline{\mathcal{H}}$ corresponds to regular behaviors (Kolmogorov, 1954). For $H(q, p) = (q^2 + 1)(p^2 + 1)/2$, shown in Fig. 9, we compare the trajectories starting from $[q(0), p(0), x(0), y(0)] = (-3, 0, -3, 0)$ calculated by the symplectic integrator (Tao, 2016) with different ω, where the calculation accuracy is second order accuracy and the time interval is 0.001. As Figs. 9(a), (b), (c), and (d) shown, the chaotic region in phase space is significantly decreasing until forming a stable limit cycle. We define $\epsilon = \|(q, p) - (x, y)\|_2$ as the calculation error of this system, shown in Figs. (e), (f), (g), and (h) that the error is decreasing with ω increasing, which fits the quantitative results of phase trajectory well.

D Other experiments

We consider the pendulum, the Lotka–Volterra, the Spring, the Hénon–Heiles, the Tao's example (Tao, 2016), the Fourier form of nonlinear Schrödinger and the vortex particle systems in our implementation. The Hamiltonian energies of these systems (except vortex particle system) are summarized as follows:

Pendulum system: $\mathcal{H}(q, p) = 3(1 - \cos(q)) + p^2$. Lotka–Volterra system: $\mathcal{H}(q, p) = p - e^p + 2q - e^q$. Spring system: $\mathcal{H}(q, p) = q^2 + p^2$. Hénon–Heiles system: $\mathcal{H}(q_1, q_2, p_1, p_2) = (p_1^2 + p_2^2)/2 + (q_1^2 + q_2^2) + (q_1^2 q_2 - q_2^3/3)/2$. Tao's example (Tao, 2016): $\mathcal{H}(q, p) = (q^2 + 1)(p^2 + 1)/2$. Fourier form of nonlinear Schrödinger equation: $\mathcal{H}(q_1, q_2, p_1, p_2) = \left[(q_1^2 + p_1^2)^2 + (q_2^2 + p_2^2)^2\right]/4 - (q_1^2 q_2^2 + p_1^2 p_2^2 - q_1^2 p_2^2 - p_1^2 q_2^2 + 4 q_1 q_2 p_1 p_2)$.

Table 2: Comparison of Prediciton error and Hamiltonian deviation between NeuralODE, HNN and NSSNN

	Prediciton error			Hamiltonian deviation		
Problems	NeuralODE	HNN	NSSNN	NeuralODE	HNN	NSSNN
Pendulum	3.4×10^{-2}	3.1×10^{-2}	$\mathbf{2.6 \times 10^{-2}}$	1.5×10^{-2}	1.3×10^{-2}	$\mathbf{7.4 \times 10^{-3}}$
Lotka-Volterra	$\mathbf{2.2 \times 10^{-2}}$	3.9×10^{-2}	2.7×10^{-2}	$\mathbf{7.4 \times 10^{-3}}$	8.8×10^{-3}	7.5×10^{-3}
Spring	2.1×10^{-2}	2.1×10^{-2}	$\mathbf{1.6 \times 10^{-2}}$	9.3×10^{-3}	$\mathbf{6.7 \times 10^{-3}}$	$\mathbf{6.7 \times 10^{-3}}$
Hénon–Heiles	1.0×10^{-1}	9.4×10^{-2}	$\mathbf{8.4 \times 10^{-2}}$	3.7×10^{-2}	4.0×10^{-2}	$\mathbf{3.5 \times 10^{-2}}$
Tao's example	3.7×10^{-2}	2.6×10^{-2}	$\mathbf{2.2 \times 10^{-2}}$	1.4×10^{-2}	1.1×10^{-2}	$\mathbf{8.2 \times 10^{-3}}$
Schrödinger	8.7×10^{-2}	5.9×10^{-2}	$\mathbf{5.7 \times 10^{-2}}$	3.8×10^{-2}	2.3×10^{-2}	$\mathbf{2.0 \times 10^{-2}}$
Vortex (2 particles)	2.1×10^{-2}	7.7×10^{-3}	$\mathbf{3.4 \times 10^{-3}}$	1.5×10^{-2}	2.8×10^{-3}	$\mathbf{2.1 \times 10^{-3}}$
Vortex (4 particles)	3.4×10^{-2}	9.4×10^{-3}	$\mathbf{6.9 \times 10^{-3}}$	8.2×10^{-2}	1.4×10^{-2}	$\mathbf{3.4 \times 10^{-3}}$

The network is trained by the dataset with noise $\sim 0.1U(-1, 1)$. The training time span, integral time step, and validation time span are 0.01, 0.01, and 0.1, respectively. Table 2 compares the Hamiltonian deviation $\epsilon_H = \|\mathcal{H}(\boldsymbol{q}_{\text{truth}}, \boldsymbol{p}_{\text{truth}}) - \mathcal{H}(\boldsymbol{q}_{\text{predict}}, \boldsymbol{p}_{\text{predict}})\|_2 / \|\mathcal{H}(\boldsymbol{q}_{\text{truth}}, \boldsymbol{p}_{\text{truth}})\|_2$ and the prediction error $\epsilon_p = \|\boldsymbol{q}_{\text{truth}} - \boldsymbol{q}_{\text{predict}}\|_1 + \|\boldsymbol{p}_{\text{truth}} - \boldsymbol{p}_{\text{predict}}\|_1$. It is clearly from the Table 2 that NSSNN either outperforms or has similar performances as NeuralODE and HNN do.