Symplectic networks: Intrinsic structure-preserving networks for identifying Hamiltonian systems

Pengzhan Jin[a,b,1], Aiqing Zhu[a,b,1], George Em Karniadakis[c], and Yifa Tang[a,b,*]

[a]LSEC, ICMSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
[b]School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China
[c]Division of Applied Mathematics, Brown University, Providence, RI 02912, USA
[1]Pengzhan Jin and Aiqing Zhu contributed equally to this work
[*]Email address: tyf@lsec.cc.ac.cn

Abstract

This work presents a framework of constructing the neural networks preserving the symplectic structure, so-called symplectic networks (SympNets). With the symplectic networks, we show some numerical results about (i) solving the Hamiltonian systems by learning abundant data points over the phase space, and (ii) predicting the phase flows by learning a series of points depending on time. All the experiments point out that the symplectic networks perform much more better than the fully-connected networks that without any prior information, especially in the task of predicting which is unable to do within the conventional numerical methods.

Key words. deep learning, physics-informed, dynamical system, Hamiltonian system, symplectic, symplectic networks

1 Introduction

Recently researchers make an effort to employ deep learning to the field of physics, such as [29], which proposes the physics-informed neural networks to solve the problems referring to various differential equations. As [29] mainly studies the nonlinear partial differential equations, [23, 31] extend it to the stochastic equations and the fractional equations. [5] adopts the variational form of PDEs and minimizes the corresponding energy functional. Factually, there are more and more works studying the applications of machine learning on complex physical systems [10, 24, 27, 28]. At the same time, there are also articles discussing the understanding of neural network from the point of view of dynamic system [4, 25]. Deep neural networks can be considered as discrete dynamical systems, the basic dynamics at each step being a linear transformation followed by a

component-wise nonlinear (activation) function. We can approximate the continuous dynamical systems based on neural network, which is in fact our main topic in this work.

It is well known that deep neural networks can approximate continuous maps [2, 12]. However, the universal approximation theorems only guarantee a small approximation error for a sufficiently large network, but do not consider the optimization and generalization errors. In order to obtain satisfactory accuracy for a given task, it requires lots of data that we cannot afford if this task is regarded as a pure approximation problem [14]. For this reason, when applying deep learning to the physical systems, the cost of data acquisition is prohibitive, and we are inevitably faced with the challenge of drawing conclusions and making decisions under partial information. Fortunately, there exists a vast amount of prior knowledge that is currently not being utilized in modern machine learning practice. Encoding such structured information into a learning algorithm results in amplifying the information content of the data that the algorithm sees, enabling it to quickly steer itself towards the right solution and generalize well even when only a few training examples are available. There have been many researches talking about how to employ the prior knowledge to construct the targeted machine learning algorithms for specific problems, where the approximated maps usually have special structures or properties which we naturally expect the trained networks possess, such as image classification [16], natural language processing [19], game playing [30], as well as the recent work [17] providing a special network structure for approximating nonlinear operators.

Turning to the main issue in this work, we are trying to explore how to impose the special structure on the neural networks for solving the corresponding problems in dynamical systems. The simplest structures that one can imagine are the Hamiltonian structure or the gradient flow structure, and this viewpoint is mentioned in the proposal on machine learning via dynamical systems [4]. Actually we will give the answer that how to construct a neural network intrinsically preserving the Hamiltonian structure, which is the core of this paper.

Denote the d-by-d identity matrix by I_d, let

$$J := \begin{pmatrix} 0 & I_d \\ -I_d & 0 \end{pmatrix},$$

which is an orthogonal, skew-symmetric real matrix, so that $J^{-1} = J^T = -J$.

Definition 1. *A matrix $H \in \mathbb{R}^{2d\times 2d}$ is called symplectic if $H^T J H = J$.*

With the concept of symplectic matrix, we can give the definition of symplectic map.

Definition 2. *A differentiable map $\Phi : U \to \mathbb{R}^{2d}$ (where $U \subset \mathbb{R}^{2d}$ is an open set) is called symplectic if the Jacobian matrix $\frac{\partial \Phi}{\partial x}$ is everywhere symplectic, i.e.,*

$$\left(\frac{\partial \Phi}{\partial x}\right)^T J \left(\frac{\partial \Phi}{\partial x}\right) = J.$$

In consideration of the Hamiltonian system

$$\begin{cases} \dot{y} = J^{-1}\nabla H(y) \\ y(t_0) = y_0 \end{cases}, \tag{1}$$

where $y(t) \in \mathbb{R}^{2d}$, and H is the Hamiltonian function representing the energy of the system (1). Let $\phi_t(y_0)$ be the phase flow of system (1). In 1899, Poincare proved that the phase flow of the Hamiltonian system is a symplectic map [11, p. 184, Theorem 2.4], i.e.,

$$\left(\frac{\partial \phi_t}{\partial y_0}\right)^T J \left(\frac{\partial \phi_t}{\partial y_0}\right) = J. \tag{2}$$

The behavior of dynamical systems at large times is a notoriously difficult problem in mathematics, particularly for discrete dynamical systems. One may encounter situations where the dynamics explodes, converges to stationary states or exhibits chaotic behavior. Fortunately, for the Hamiltonian system, these problems can be alleviated by imposing the symplectic structure on the numerical methods due to (2). There are some well-developed works on symplectic integration, see for examples [7, 8, 11, 18]. As the symplectic numerical integrators yield transformative results across diverse applications based on the Hamiltonian systems [6, 22, 26, 32], we have to consider the construction of the symplectic networks and how it impacts on the numerical methods for the Hamiltonian systems.

The remaining parts of this paper are organized as follows. Section 2 briefly summarizes the main problem we will solve. The detailed process of constructing the symplectic networks and some related results are shown in Section 3. In Section 4, we give the unit triangular factorization of the matrix symplectic group, which is necessary to the construction of the symplectic networks. Section 5 presents the numerical results of solving and predicting the phase flows of the Hamiltonian systems. Some conclusions will be given in the last section.

2 Problem setup

In this work we focus on the Hamiltonian system (1). Since the equation (2) holds, it is natural to search for numerical methods that share this property. Different from the conventional symplectic numerical integrators, we will use deep neural networks instead, to approximate the phase flow ϕ_t for fixed t. Crucially, we expect the networks to be intrinsically symplectic, by imposing special structure on it. The main task we are aiming to complete is just to design the symplectic networks, subsequently it can be applied to solving or predicting the phase flows of the Hamiltonian systems accurately and efficiently.

3 Neural networks for identifying phase flow of Hamiltonian system

The conventional numerical method for solving Hamiltonian system is to construct a special scheme that preserves the symplectic structure which is the intrinsic nature of the phase flow of Hamiltonian system, and the designed scheme is subsequently used to calculate the phase points step by step. The difficulty of above method is that a symplectic scheme is often an implicit scheme, which may be computationally expensive, especially for high-order ones. Here we exploit the deep learning algorithm leading to an explicit expression, to obtain the numerical phase flow of Hamiltonian system. Furthermore, a new targeted network structure is proposed to this problem.

3.1 Networks applying structure information

There have been some works studying how to apply deep learning to solving differential equations, i.e., physics-informed neural networks. Instead of using the general physics-informed networks to approximate the solution of (1) over the whole time interval, here we learn the phase flow $\phi_h(y_0)$ of (1) to calculate the solutions. Same to what numerical integrators do, the trained network is used to compute the phase point after time step h of the start point y_0, i.e., the input is phase point y_0 while the output is the phase point $y_1 = \phi_h(y_0)$.

Assume that the phase flows of (1) referred to are constrained in a compact space V. We first choose many finite phase points from V, denoted by $\{x_i\}_1^N$, then obtain the phase points $\{y_i = \phi_h(x_i)\}_1^N$ of next time step of $\{x_i\}_1^N$ pointwise by a high-order symplectic scheme, such as symplectic Runge–Kutta method [11, Chapter VI.4]. Naturally,

$$\mathcal{T} = \{(x_i, y_i)\}_1^N \tag{3}$$

is viewed as the training set for learning, where $y_i = \phi_h(x_i)$. The neural network Φ_h as numerical integrator can be learned by minimizing the mean squared error loss

$$MSE = MSE_d + w \cdot MSE_s, \tag{4}$$

where

$$MSE_d = \frac{1}{N}\sum_{i=1}^{N} \|\Phi_h(x_i) - y_i\|^2$$

is the loss with respect to the data points, and

$$MSE_s = \frac{1}{N}\sum_{i=1}^{N} \left\| \left[\left(\frac{\partial \Phi_h}{\partial x}\right)^T J \left(\frac{\partial \Phi_h}{\partial x}\right) \right](x_i) - J \right\|_F^2 \tag{5}$$

is the loss to preserve the geometrical structure of Hamiltonian system, i.e., symplectic structure. Note that w is the weight to set.

Suppose that there exist a small $\epsilon > 0$ and $\delta(x)$ such that

$$\Phi_h(x) - \phi_h(x) = \epsilon\delta(x)$$

where $\|\delta(x)\| \leq 1$, and $\|\frac{\partial \delta}{\partial x}\|$ is bounded due to [13]. Then for any x_i, there holds

$$\begin{aligned} \left\| \left(\frac{\partial \Phi_h}{\partial x}\right)^T J \left(\frac{\partial \Phi_h}{\partial x}\right)(x_i) - J \right\|_F &= \left\| \epsilon \left[\left(\frac{\partial \delta}{\partial x}\right)^T J \left(\frac{\partial \phi_h}{\partial x}\right) + \left(\frac{\partial \phi_h}{\partial x}\right)^T J \left(\frac{\partial \delta}{\partial x}\right) \right](x_i) + O(\epsilon^2) \right\|_F \\ &= O(\epsilon) \end{aligned}$$

which implies that

$$MSE_s = O(\epsilon^2) \approx O(MSE_d).$$

The above discussion shows that even though the value of w is set to 0, the MSE_s will tend to 0 when MSE_d tends to 0, and therefore to control the structure information, the MSE_s may be not required. If we enforce a small MSE_s to improve the error, the weight w should be set to a large value, however, which will severely slow down the convergence. In experiments, it is expensive to obtain a satisfactory optimized network with large w, thus we just set the w to 0 in the section of numerical results.

3.2 Symplectic networks (SympNets)

The previous general networks preserve the symplectic structure by controlling the loss (maybe encoding symplecticity condition as (5)) on data points, however, the weakness is obvious that the trained networks are hard to keep structure over the whole phase space V, especially when the sampled points are sparse in a high-dimensional V. For the purpose of making our trained networks hold symplectic structure more extensively and intrinsically, we tend to bring the prior information into the network structure like what deep learning researchers prefer to do.

Suppose that Φ_1 and Φ_2 are two symplectic maps, then it is easy to show that their composite map $\Phi_2 \circ \Phi_1$ is also symplectic due to the chain rule. Thus the symplectic maps are closed under the composite operation. Aim to construct the destination symplectic map, we make an effort to find the simplest linear/nonlinear symplectic map as the unit layer of the network, which is similar to the general fully-connected neural network. Hence the network yielded will stay symplectic due to the closure of composite operation. It is noteworthy that the unit symplectic map should be simple enough so that it can be freely parameterized, since the now available optimization techniques in deep learning focus on the unconstrained problems.

In reference to the linear symplectic unit, let

$$\ell_{up}(x) = \begin{pmatrix} I & S \\ 0 & I \end{pmatrix} x + b, \quad \ell_{low}(x) = \begin{pmatrix} I & 0 \\ S & I \end{pmatrix} x + b, \quad x, b \in \mathbb{R}^{2d} \tag{6}$$

where $S \in \mathbb{R}^{d \times d}$ is symmetric. Obviously, ℓ_{up} and ℓ_{low} are linear and symplectic, however, they are too simple to express a general linear symplectic map. In order to strengthen the expressivity of linear layer, we compound several ℓ_{up} and ℓ_{low} alternately as

$$\mathcal{L}_n(x) = \begin{pmatrix} I & 0/S_n \\ S_n/0 & I \end{pmatrix} \cdots \begin{pmatrix} I & 0 \\ S_2 & I \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} x + b, \quad x, b \in \mathbb{R}^{2d}, \tag{7}$$

hence $\mathcal{L}_n$ is viewed as the linear unit in our symplectic network. For the sake of convenience, we can replace S_i with $(S_i + S_i^T)/2$ to eliminate the constraint on the symmetry in practice. Now a essential problem is raised naturally, that, are maps like $\mathcal{L}_n$ able to represent any linear symplectic map? The answer is yes and we will show details in Section 4.

Nevertheless, it is unclear that how to search for a concise nonlinear expression which satisfies symplecticity condition and can be computed efficiently simultaneously. Here is a theorem providing a method about constructing symplectic maps [11, Chapter VI.5].

Theorem 1. *For any smooth function $S(P, q) : U \times V \to \mathbb{R}$ $(U, V \subset \mathbb{R}^d)$,*

$$Q = \frac{\partial S}{\partial P}(P, q), \qquad p = \frac{\partial S}{\partial q}(P, q)$$

locally define a symplectic map $\Phi : (p, q) \to (P, Q)$ if $\frac{\partial^2 S}{\partial P \partial q}$ is invertible.

Proof. The proof can be found in [11, p. 197] □

Now we are able to generate symplectic maps by Theorem 1. Think about the smooth activation function σ, such as sigmoid, and let

$$S(P, q) = \sum_{i=1}^{d} (\int \sigma)(P_i) + q^T P,$$

where $\int \sigma$ is the antiderivative of σ and P_i is the i-th component of P. We subsequently derive that

$$\begin{pmatrix} P \\ Q \end{pmatrix} = \Phi \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p \\ \sigma(p) + q \end{pmatrix}$$

is a symplectic map, where $\sigma(p) = (\sigma(p_1), \cdots, \sigma(p_d))^T$. For notational convenience, we denote this Φ by

$$\mathcal{N} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & 0 \\ \sigma & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p \\ \sigma(p) + q \end{pmatrix},$$

here I and 0 are viewed as identity and zero maps, respectively. Thus $\mathcal{N}$ is obtained as the nonlinear symplectic layer. Similar to (6), we define

$$\mathcal{N}_{up} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & \sigma \\ 0 & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix}, \quad \mathcal{N}_{low} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & 0 \\ \sigma & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix}.$$

With linear and nonlinear units $\mathcal{L}_n$ and $\mathcal{N}_{up/low}$, we construct the multi-layer feed-forward neural network with symplectic structure:

$$\Phi = \mathcal{L}_n \circ \mathcal{N}_{up/low} \circ \mathcal{L}_n \circ \cdots \circ \mathcal{N}_{up} \circ \mathcal{L}_n \circ \mathcal{N}_{low} \circ \mathcal{L}_n = \mathcal{L}_n \circ (\mathcal{N}_{up/low} \circ \mathcal{L}_n)^k.$$

Note that $\mathcal{N}_{up}$ and $\mathcal{N}_{low}$ appear alternately, and $\mathcal{L}_n$ in different layers have different parameters to be optimized. Furthermore, taking into account the fact that, for a small time step h the numerical integrator should be close to the identity map, it is necessary to modify Φ to be consistent with the behavior of tiny h. We define

$$\mathcal{L}_n^h(x) = \begin{pmatrix} I & 0/h \cdot S_n \\ h \cdot S_n/0 & I \end{pmatrix} \cdots \begin{pmatrix} I & 0 \\ h \cdot S_2 & I \end{pmatrix} \begin{pmatrix} I & h \cdot S_1 \\ 0 & I \end{pmatrix} x + h \cdot b, \quad x, b \in \mathbb{R}^{2d},$$

and

$$\mathcal{N}_{up}^h \begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & h \cdot \sigma \\ 0 & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix}, \quad \mathcal{N}_{low}^h \begin{pmatrix} p \\ q \end{pmatrix} = \begin{bmatrix} I & 0 \\ h \cdot \sigma & I \end{bmatrix} \begin{pmatrix} p \\ q \end{pmatrix},$$

consequently

$$\Phi_h = \mathcal{L}_n^h \circ \mathcal{N}_{up/low}^h \circ \mathcal{L}_n^h \circ \cdots \circ \mathcal{N}_{up}^h \circ \mathcal{L}_n^h \circ \mathcal{N}_{low}^h \circ \mathcal{L}_n^h = \mathcal{L}_n^h \circ (\mathcal{N}_{up/low}^h \circ \mathcal{L}_n^h)^k$$

satisfies $\Phi_0 = I$ in the sense of map. Here Φ_h is regarded as a k layers **symplectic network (SympNet)** with n sub-layers.

For the same training set $\mathcal{T}$ defined in (3), the loss of neural network Φ_h is defined by

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \|\Phi_h(x_i) - y_i\|^2.$$

This way embeds the geometrical information in its network structure rather than the loss, compared to (4). What is much stronger than the previous loss-controlled structure-preserving network is that symplectic network keeps symplectic accurately everywhere, even there is no data point. Therefore symplectic networks are possible to predict the phase flows in unknown area corresponding to training data. We will show some numerical results in Section 5.

3.3 Extension of SympNets

The SympNets described in the previous subsection will be used in our numerical experiments, nevertheless, some setups on the nets are unnecessary to preserve the structure or guarantee the expressivity, such as the alternation of $\mathcal{N}_{up}$ and $\mathcal{N}_{low}$. Here we extend the aforementioned symplectic networks to a more general form.

Denote

$$\mathcal{V}_n = \left\{ f \middle| f(x) = \begin{pmatrix} I & 0/S_n \\ S_n/0 & I \end{pmatrix} \cdots \begin{pmatrix} I & 0 \\ S_2 & I \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} x + b, S_i \in \mathbb{R}^{d\times d}, S_i^T = S_i, b \in \mathbb{R}^{2d} \right\}$$
$$\bigcup \left\{ f \middle| f(x) = \begin{pmatrix} I & S_n/0 \\ 0/S_n & I \end{pmatrix} \cdots \begin{pmatrix} I & S_2 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ S_1 & I \end{pmatrix} x + b, S_i \in \mathbb{R}^{d\times d}, S_i^T = S_i, b \in \mathbb{R}^{2d} \right\},$$

$$\mathcal{V} = \bigcup_{n=1}^{\infty} \mathcal{V}_n,$$

and

$$\mathcal{W} = \left\{ f : \mathbb{R}^{2d} \to \mathbb{R}^{2d} \middle| f = \begin{bmatrix} I & h\cdot\sigma \\ 0 & I \end{bmatrix}, h \in \mathbb{R} \right\} \bigcup \left\{ f : \mathbb{R}^{2d} \to \mathbb{R}^{2d} \middle| f = \begin{bmatrix} I & 0 \\ h\cdot\sigma & I \end{bmatrix}, h \in \mathbb{R} \right\},$$

where σ is a fixed activation function. In fact, Section 4 points out that $\mathcal{V}$ is equivalent to the set consisting of all the linear symplectic maps. Based on $\mathcal{V}$ and $\mathcal{W}$, we have the following definition.

Definition 3. *For $\{v_i\}_1^k \cup \{v_{out}\} \subset \mathcal{V}$, $\{w_i\}_1^k \subset \mathcal{W}$, let*

$$\psi = v_{out} \circ w_k \circ v_k \circ \cdots \circ w_1 \circ v_1.$$

*ψ is called the **general symplectic network**. Futhermore, we define the collection of general symplectic networks as*

$$\Psi = \{\psi | \psi \text{ is a general symplectic network}\}.$$

Now we do not require the linear units to start with ℓ_{up} either the nonlinear units to appear alternatively. Based on this setup, the set Ψ surprisingly forms a group in the sense of compound.

Theorem 2 (Algebraic Structure)**.** *The collection of the general symplectic networks Ψ is a group.*

Proof. We have shown that Φ_0 is the identity map which belongs to Ψ. Moreover, the associative law and the closure obviously hold by the definition of Ψ. What we need to confirm is there exists an inverse element for any ψ, i.e., $\psi^{-1} \in \Psi$. According to the observation that

$$\begin{pmatrix} I & S \\ 0 & I \end{pmatrix}^{-1} = \begin{pmatrix} I & -S \\ 0 & I \end{pmatrix}, \quad \begin{pmatrix} I & 0 \\ S & I \end{pmatrix}^{-1} = \begin{pmatrix} I & 0 \\ -S & I \end{pmatrix} \tag{8}$$

as well as

$$\begin{bmatrix} I & h\cdot\sigma \\ 0 & I \end{bmatrix}^{-1} = \begin{bmatrix} I & -h\cdot\sigma \\ 0 & I \end{bmatrix}, \quad \begin{bmatrix} I & 0 \\ h\cdot\sigma & I \end{bmatrix}^{-1} = \begin{bmatrix} I & 0 \\ -h\cdot\sigma & I \end{bmatrix}, \tag{9}$$

we derive that

$$\psi^{-1} = v_1^{-1} \circ w_1^{-1} \circ \cdots \circ v_k^{-1} \circ w_k^{-1} \circ v_{out}^{-1} \in \Psi.$$

Therefore Ψ is a group. □

In Section 3.2, a detailed structure of SympNets, i.e., Φ_h, is provided for identifying the Hamiltonian systems. Now what we expect is that, to go a step further, Φ_h becomes a symmetric integrator, which satisfies $\Phi_h^{-1} = \Phi_{-h}$. By the extension of SympNets, we can construct the targeted integrator under the Ψ as $\widetilde{\Phi}_h = \Phi_{-h}^{-1} \circ \Phi_h$, then

$$\widetilde{\Phi}_h^{-1} = (\Phi_{-h}^{-1} \circ \Phi_h)^{-1} = \Phi_h^{-1} \circ \Phi_{-h} = \widetilde{\Phi}_{-h}.$$

In practice, we may firstly obtain the expression of Φ_{-h}^{-1} due to (8) and (9), then train $\widetilde{\Phi}_h = \Phi_{-h}^{-1} \circ \Phi_h$ on the data and finally go for the evaluation. It is noteworthy that Φ_h and Φ_{-h}^{-1} share the same parameters during training. The study of the effectiveness of this technique and how it impacts on the accuracy will be left to the future, in this paper we only apply the structure shown in Section 3.2.

4 Parametrization of the matrix symplectic group

Since the current optimization in deep learning focuses on the unconstrained problems, it is necessary to find a representation for symplectic matrix which can be freely parameterized. [1, 3, 9, 20, 21] provide several methods for the parametrization of the matrix symplectic group, nevertheless, all of its representation requires permutation matrices and nonsingular matrices which are hard to be applied to neural networks, here we are searching for a more effective way. In consideration of that unit triangular symplectic matrices

$$\begin{pmatrix} I & S \\ 0 & I \end{pmatrix}, \quad \begin{pmatrix} I & 0 \\ S & I \end{pmatrix} \quad (S^T = S)$$

can be parameterized as

$$\begin{pmatrix} I & (S+S^T)/2 \\ 0 & I \end{pmatrix}, \quad \begin{pmatrix} I & 0 \\ (S+S^T)/2 & I \end{pmatrix}$$

where S is a matrix without any constraint, we are tending to decompose a general symplectic matrix into several simple symplectic matrices of above types.

Denote

$$SP = \{H \in \mathbb{R}^{2d\times 2d} | H^T J H = J\},$$

$$L_n = \left\{ \begin{pmatrix} I & 0/S_n \\ S_n/0 & I \end{pmatrix} \cdots \begin{pmatrix} I & 0 \\ S_2 & I \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \middle| S_i \in \mathbb{R}^{d\times d}, S_i^T = S_i, i = 1, 2, \cdots, n \right\},$$

where the unit upper triangular symplectic matrices and the unit lower triangular symplectic matrices appear alternately. And it is clear that $L_m \subset L_n \subset SP$ for all integers $1 \leq m \leq n$. Now the main theorem is given as following.

Theorem 3 (Unit Triangular Factorization)**.** *$SP = L_9$.*

Proof. The proof can be found in our previous work [15]. □

In [15], we systematically present several existing modern factorizations of the matrix symplecic group, and propose the unit triangular factorization described as Theorem 3. This factorization

induces the unconstrained parametrization of the matrix symplectic group, by replacing the S_i with $(S_i + S_i^T)/2$. It enables us to make use of the symplectic matrix as a module in a deep neural network, just like what we are doing. Furthermore, [15] provides more unconstrained parametrization for the structured subsets of the matrix symplectic group, such as the positive definite symplectic matrices and the singular symplectic matrices, which may be applied to the problems with these constraints.

5 Numerical results

In this section, we check the above two types of networks in: (1) solving Hamiltonian system by learning abundant data over phase space, (2) predicting phase flow given a series of phase points depending on time.

5.1 Solving systems

To solve the systems by learning phase flow ϕ_h , we need to sample lots of data points from phase space to provide sufficient information about how ϕ_h acts on the whole space. After learning, we apply the trained network Φ_h to generating phase flow step by step, subsequently to compare with the true flow. We will use the two types of networks: the general fully-connected networks(FNNs) and the proposed symplectic networks(SympNets). Table 1 shows the default hyper-parameters we set.

Type	Structure	Activation	Optimizer	Learning Rate	Epochs
FNN	$[2d, 50, 50, 2d]$	Sigmoid	Adam	0.1/0.01	1e6
SympNet	8 layers with 5 sub-layers	Sigmoid	Adam	0.1/0.01	1e6

Table 1: **Default hyper-parameters.** Learning rate is set to 0.1 for solving, and 0.01 for predicting.

5.1.1 Pendulum

Here we solve the mathematical pendulum (mass $m = 1$, massless rod of length $l = 1$, gravitational acceleration $g = 1$) which is a system with one degree of freedom having the Hamiltonian

$$H(p, q) = \frac{1}{2}p^2 - \cos(q).$$

We set $h = 0.1$, and generate 10000 training/test data points randomly from compact set $[-\sqrt{2}, \sqrt{2}] \times [-\pi/2, \pi/2]$. After training, we use the trained network to compute three flows starting at $(0, 0.5)$, $(0, 1.0)$, $(0, 1.5)$ for 1000 steps, respectively.

The training MSE and the test MSE are shown in Table 2. Figure 1 shows the numerical flows solved by FNN and SympNet, and because of the structure information on SympNet, the flows by SympNet keep the true trajectories all the time while the flows by FNN deviate from the true trajectories over time. The trajectories of q are given in Figure 2. We can see that SympNet

provides a more reliable result than FNN in long time solution. Furthermore, SympNet preserves the energy nearly, hence keeps the peak/valley values of q invariant.

Type	Pendulum		Lotka-Volterra	
	Training MSE	Test MSE	Training MSE	Test MSE
FNN	3.14e-7	3.22e-7	5.94e-7	6.23e-7
SympNet	2.88e-7	2.74e-7	1.97e-5	1.82e-5

Table 2: **MSEs of FNN and SympNet on solving pendulum system and Lotka-Volterra system.**

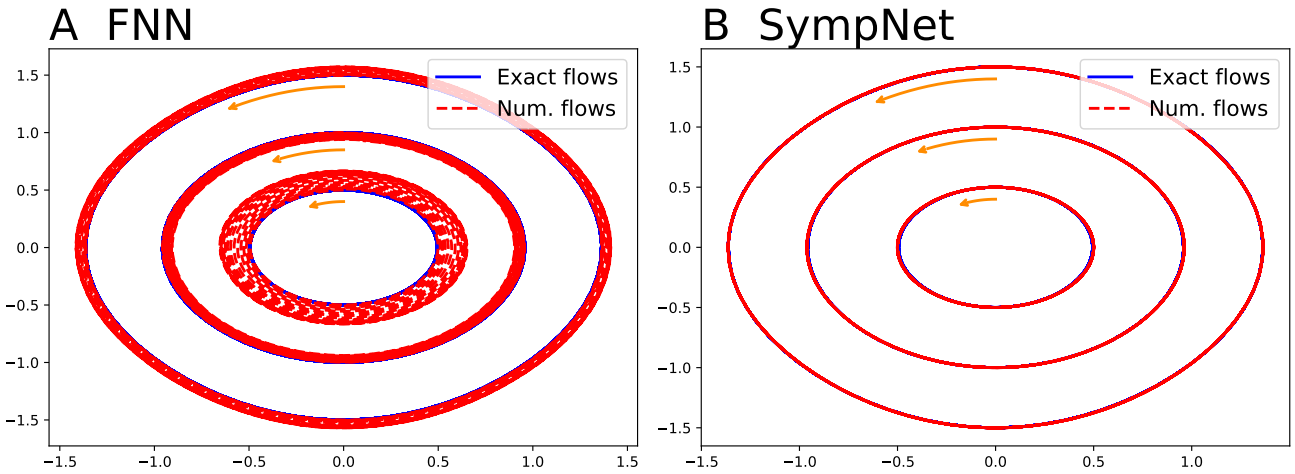

Figure 1: **Numerical flows solved by FNN and SympNet for pendulum.** (**A**) represents the flows solved by FNN, the flows will deviate from the true trajectories over time. (**B**) represents the flows solved by SympNet, the flows keep the true trajectories all the time.

5.1.2 Lotka-Volterra

Here we solve the Lotka-Volterra system in logarithmic scale which is a system with one degree of freedom having the canonically Hamiltonian

$$H(p,q) = p - e^p + 2q - e^q.$$

We set $h = 0.1$, and generate 10000 training/test data points randomly from compact set $[-2, 1.5] \times [-0.5, 2]$. After training, we use the trained network to compute three flows starting at $(0, 1.0)$, $(0, 1.25)$, $(0, 1.5)$ for 1000 steps, respectively.

The training MSE and the test MSE are shown in Table 2. Figure 3 shows the numerical flows solved by FNN and SympNet, and it reflects the superiority of SympNet again, which is similar to the case of pendulum, i.e., the SympNet preserves the geometrical structure of numerical phase flows. However, from Table 2 we know that the training MSE of SympNet is much more larger than of FNN in this case, which indicates that SympNets are not easy to train. It is worth noting that the SympNet used has only 63 parameters while FNN has thousands, moreover, 8 activation layers

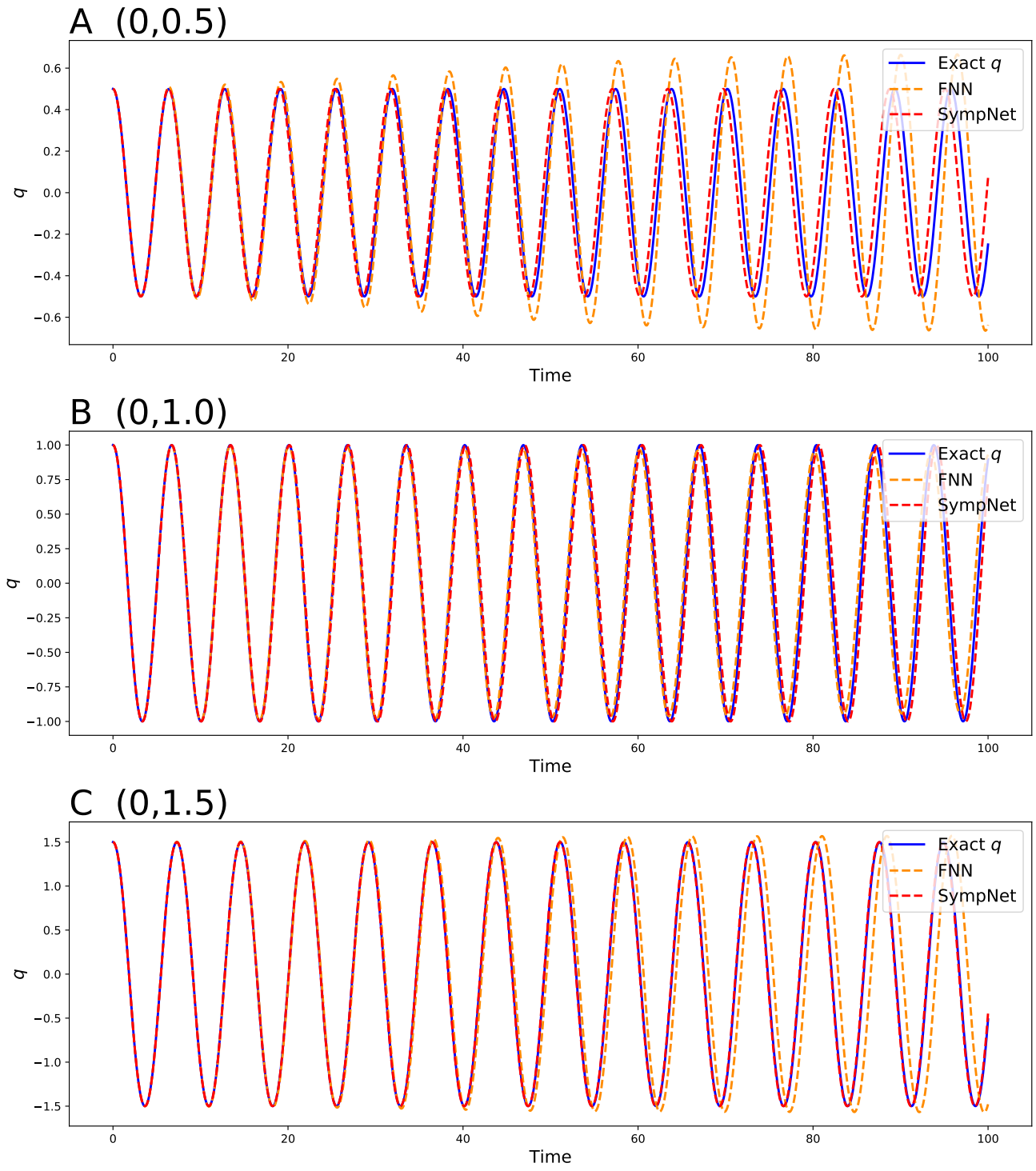

Figure 2: **Numerical q solved by FNN and SympNet for pendulum.** (**A**), (**B**), (**C**) represent the time based trajectories of q starting from $(0, 0.5)$, $(0, 1.0)$, $(0, 1.5)$ respectively.

increase the difficulty of learning due to the disappearance of gradient. Since the wide difference in training MSE, it is meaningless to compare the trajectories of q by FNN and SympNet. How to enhance the approximation capability of SympNets is till a problem.

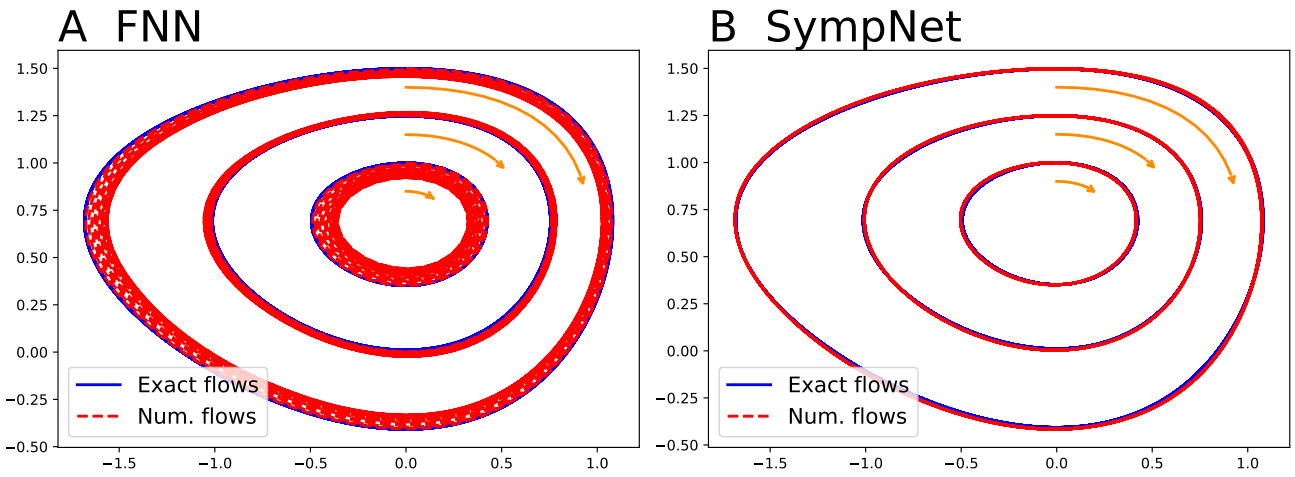

Figure 3: **Numerical flows solved by FNN and SympNet for Lotka-Volterra.** (**A**) represents the flows solved by FNN, the flows will deviate from the true trajectories over time. (**B**) represents the flows solved by SympNet, the flows keep the true trajectories all the time.

5.2 Predicting systems

Now we consider a new task that conventional numerical methods are unable to achieve. For an unknown Hamiltonian system, i.e., the Hamiltonian $H(p,q)$ is unknown, we try to predict the flow based on gathering a series of phase points $\{x_i\}_0^n$ with time step h. Then $\mathcal{T} = \{(x_{i-1}, x_i)\}_1^n$ is viewed as the training data. We apply neural networks to learning $\mathcal{T}$ and giving the predicted phase flow starting at the rear x_n. Table 1 shows the default hyper-parameters we set.

5.2.1 Pendulum

For pendulum system which has the Hamiltonian

$$H(p,q) = \frac{1}{2}p^2 - \cos(q),$$

we first obtain the flow starting from $(0, 1.0)$ with 40 points and time step 0.1 as the training data, i.e., $\mathcal{T} = \{(x_{i-1}, x_i)\}_1^n$ where $n = 40$, $x_0 = (0, 1.0)$ as well as

$$x_i = \phi_h(x_{i-1}), \quad h = 0.1, \quad i = 1, 2, \cdots, n.$$

After training on $\mathcal{T}$, we use the trained network Φ_h to compute the flow starting at x_n for 1000 steps.

Figure 4 shows the predicted flows by FNN and SympNet. We find that SympNet discovers the unknown trajectory successfully while FNN completely fail to do that. Figure 5 provides the detailed trajectories of q by FNN and SympNet, where SympNet gives comparatively accurate numerical result in the previous cycles, while values of q by FNN stay invariant during predicting.

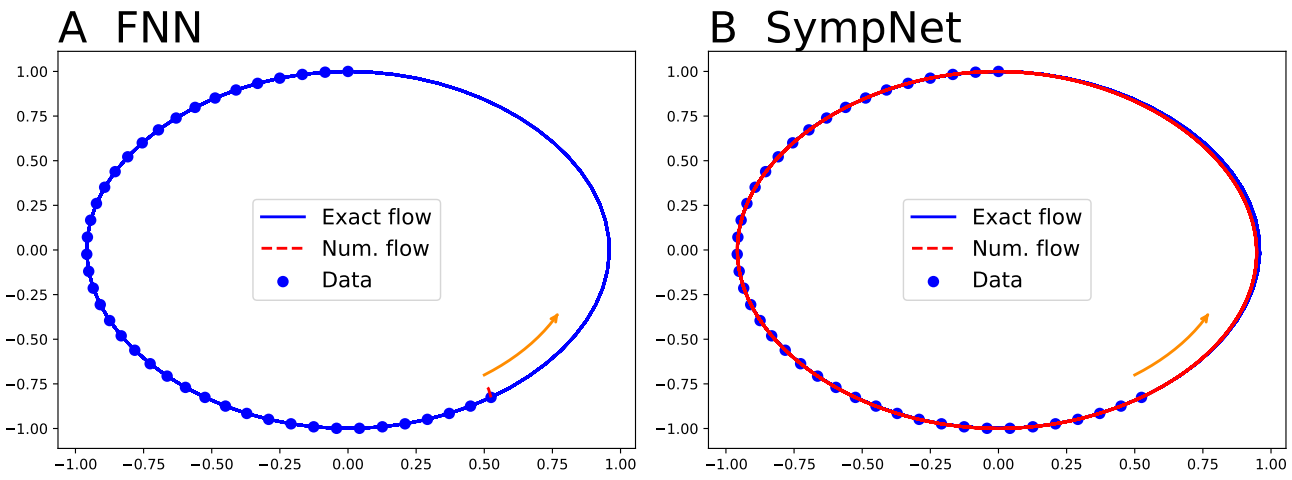

Figure 4: **Numerical flows predicted by FNN and SympNet for pendulum.** SympNet discovers the unknown trajectory successfully while FNN completely fail to do that.

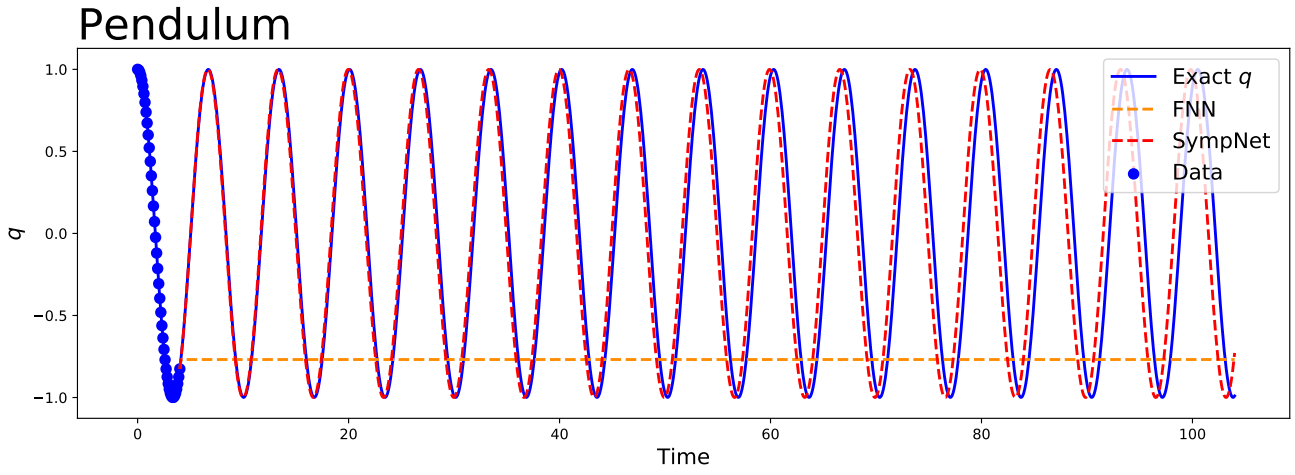

Figure 5: **Numerical q predicted by FNN and SympNet for pendulum.** SympNet gives comparatively accurate numerical result in the previous cycles, while values of q by FNN stay invariant during predicting.

5.2.2 Lotka-Volterra

For Lotka-Volterra system which has the Hamiltonian

$$H(p,q) = p - e^p + 2q - e^q,$$

we first obtain the flow starting from $(0, 1.0)$ with 25 points and time step 0.1 as the training data, i.e., $\mathcal{T} = \{(x_{i-1}, x_i)\}_1^n$ where $n = 25$, $x_0 = (0, 1.0)$ as well as

$$x_i = \phi_h(x_{i-1}), \quad h = 0.1, \quad i = 1, 2, \cdots, n.$$

After training on $\mathcal{T}$, we use the trained network Φ_h to compute the flow starting at x_n for 1000 steps.

Figure 6 shows the predicted flows by FNN and SympNet and Figure 7 provides the detailed trajectories of q. Same to the case of pendulum, SympNet appears excellent on predicting while FNN is unable to provide any available information about the unknown trajectory.

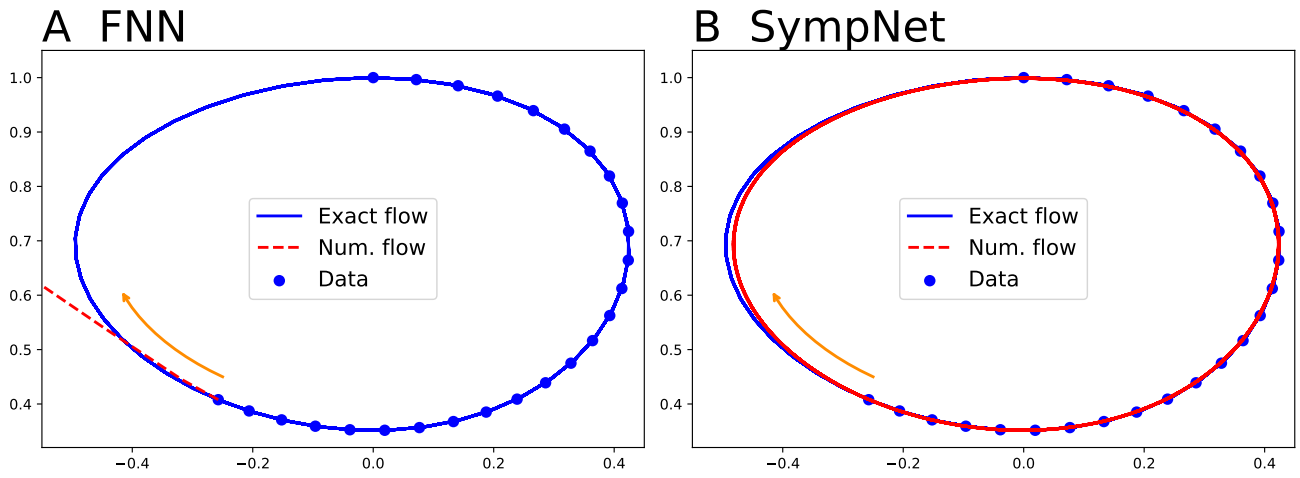

Figure 6: **Numerical flows predicted by FNN and SympNet for Lotka-Volterra.** SympNet discovers the unknown trajectory successfully while FNN completely fail to do that.

5.2.3 Kepler problem

Now we consider a four-dimensional system, the Kepler problem(mass $m = 1$, $M = 1$, gravitational constant=1), which has the Hamiltonian

$$H(\mathbf{p}, \mathbf{q}) = H(p_1, p_2, q_1, q_2) = \frac{1}{2}(p_1^2 + p_2^2) - \frac{1}{\sqrt{q_1^2 + q_2^2}},$$

we first obtain the flow starting from $(1, 0, 0, 1)$ with 40 points and time step 0.1 as the training data, i.e., $\mathcal{T} = \{(x_{i-1}, x_i)\}_1^n$ where $n = 40$, $x_0 = (1, 0, 0, 1)$ as well as

$$x_i = \phi_h(x_{i-1}), \quad h = 0.1, \quad i = 1, 2, \cdots, n.$$

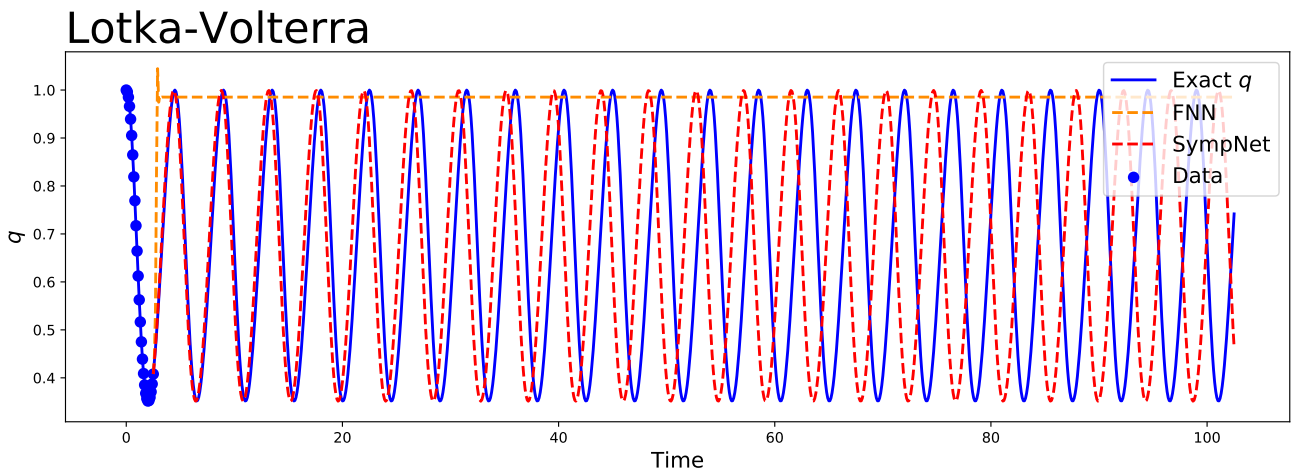

Figure 7: **Numerical q predicted by FNN and SympNet for Lotka-Volterra.** SympNet gives comparatively accurate numerical result in the previous cycles, while values of q by FNN stay invariant after several steps during predicting.

After training on $\mathcal{T}$, we use the trained network Φ_h to compute the flow starting at x_n for 1000 steps.

In this case, we only study the behavior of SympNet on predicting, since the previous cases have pointed out the ineffectiveness of FNN. Figure 8 shows the numerical flows predicted by SympNet, where both the trajectories of velocity $\mathbf{p}$ and position $\mathbf{q}$ are almost consistent with the true trajectories. Figure 9 provides the detailed trajectories of q_1 and q_2. Different from solving, the task for predicting requires only little data points, thus the scale of training set needed will remain acceptable even in high-dimensional problems.

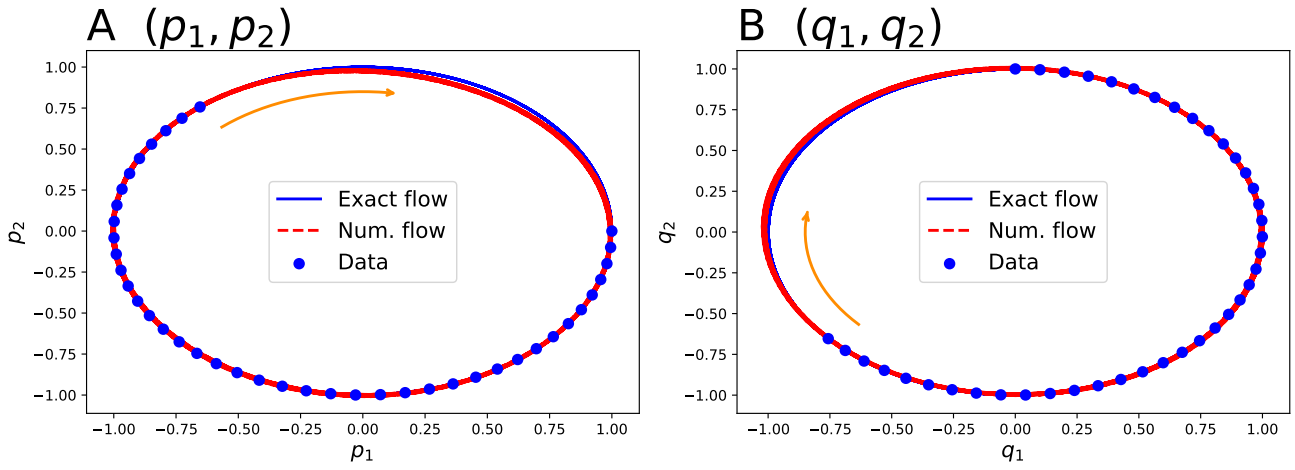

Figure 8: **Numerical flows predicted by SympNet for Kepler problem.** (**A**) represents the predicted flows of velocity $\mathbf{p}$, and (**B**) represents the predicted flows of position $\mathbf{q}$. Both the trajectories of $\mathbf{p}$ and $\mathbf{q}$ are almost consistent with the true trajectories.

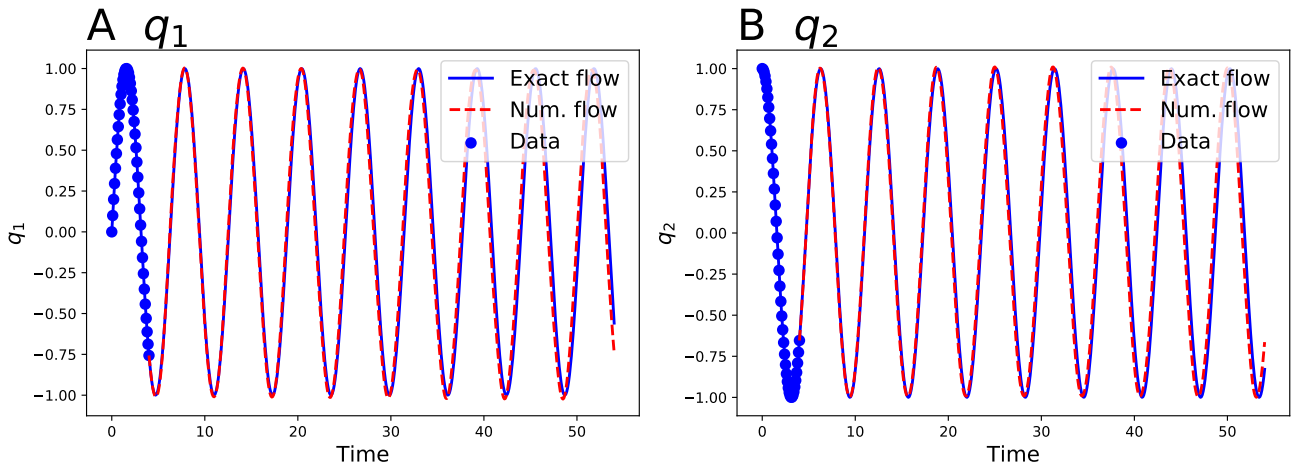

Figure 9: **Numerical q_1, q_2 predicted by SympNet for Kepler problem.** The numerical results are comparatively accurate in the previous cycles.

6 Conclusions

This work presents a framework of constructing the neural networks preserving the symplectic structure. The key of the construction is the unit triangular factorization of the matrix symplectic group, which has been proposed in our previous work [15]. Furthermore, the general symplectic networks shown in Definition 3 are provided with an algebraic structure, which form a group in fact. This algebraic structure indicates the possibility of building the more efficient symplectic networks, such as the symmetric symplectic networks.

With the symplectic networks, we show some numerical results about solving the Hamiltonian systems by learning abundant data points over the phase space, and predicting the phase flows by learning a series of points depending on time. All the experiments point out that the symplectic networks perform much more better than the fully-connected networks that are without any prior information, especially in the task of predicting which is unable to do within the conventional numerical methods.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant No. 11771438), and Major Project on New Generation of Artificial Intelligence from MOST of China (Grant No. 2018AAA0101002).

References

[1] A. Bunse-Gerstner. Matrix factorizations for symplectic qr-like methods. *Linear Algebra and its Applications*, 83:49–77, 1986.

[2] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[3] F. M. Dopico and C. R. Johnson. Parametrization of the matrix symplectic group and applications. *SIAM Journal on Matrix Analysis and Applications*, 31(2):650–673, 2009.

[4] W. E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[5] W. E and B. Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[6] E. Faou, V. Gradinaru, and C. Lubich. Computing semiclassical quantum dynamics with hagedorn wavepackets. *SIAM Journal on Scientific Computing*, 31(4):3027–3041, 2009.

[7] K. Feng. On difference schemes and symplectic geometry. In *Proceedings of the 5th international symposium on differential geometry and differential equations*, 1984.

[8] K. Feng. *Collected Works of Feng Kang (II)*. National Defense Industry Press, 1995.

[9] U. Flaschka, V. Mehrmann, and D. Zywietz. An analysis of structure preserving methods for symplectic eigenvalue problems, rairo automat. *Prod. Inform. Ind*, 25:f1991, 1991.

[10] M. Gulian, M. Raissi, P. Perdikaris, and G. Karniadakis. Machine learning of space-fractional differential equations. *SIAM Journal on Scientific Computing*, 41(4):A2485–A2509, 2019.

[11] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[12] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[13] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

[14] P. Jin, L. Lu, Y. Tang, and G. E. Karniadakis. Quantifying the generalization error in deep learning in terms of data distribution and neural network smoothness. *arXiv preprint arXiv:1905.11427*, 2019.

[15] P. Jin, Y. Tang, and A. Zhu. Unit triangular factorization of the matrix symplectic group. *arXiv preprint arXiv:1912.10926*, 2019.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] L. Lu, P. Jin, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[18] C. Lubich. *From quantum to classical molecular dynamics: reduced models and numerical analysis*. European Mathematical Society, 2008.

[19] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[20] D. S. Mackey and N. Mackey. *On the determinant of symplectic matrices*. Manchester Centre for Computational Mathematics, 2003.

[21] V. Mehrmann. A symplectic orthogonal method for single input or single output discrete time optimal quadratic control problems. *SIAM Journal on Matrix Analysis and Applications*, 9(2):221–247, 1988.

[22] I. Omelyan, I. Mryglod, and R. Folk. Symplectic analytically integrable decomposition algorithms: classification, derivation, and application to molecular dynamics, quantum and celestial mechanics simulations. *Computer Physics Communications*, 151(3):272–314, 2003.

[23] G. Pang, L. Lu, and G. E. Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[24] G. Pang, L. Yang, and G. E. Karniadakis. Neural-net-induced gaussian process regression for function approximation and pde solution. *Journal of Computational Physics*, 384:270–288, 2019.

[25] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[26] H. Qin, J. Liu, J. Xiao, R. Zhang, Y. He, Y. Wang, Y. Sun, J. W. Burby, L. Ellison, and Y. Zhou. Canonical symplectic particle-in-cell method for long-term large-scale simulations of the vlasov–maxwell equations. *Nuclear Fusion*, 56(1):014001, 2015.

[27] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[28] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational Physics*, 335:736–746, 2017.

[29] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[30] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[31] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[32] R. Zhang, J. Liu, Y. Tang, H. Qin, J. Xiao, and B. Zhu. Canonicalization and symplectic simulation of the gyrocenter dynamics in time-independent magnetic fields. *Physics of Plasmas*, 21(3):032504, 2014.