Learning Poisson Systems and Trajectories of Autonomous Systems via Poisson Neural Networks

Pengzhan Jin, Zhen Zhang, Ioannis G. Kevrekidis, and George Em Karniadakis

***Abstract*— We propose the Poisson neural networks (PNNs) to learn Poisson systems and trajectories of autonomous systems from data. Based on the Darboux–Lie theorem, the phase flow of a Poisson system can be written as the composition of: 1) a coordinate transformation; 2) an extended symplectic map; and 3) the inverse of the transformation. In this work, we extend this result to the unknotted trajectories of autonomous systems. We employ structured neural networks with physical priors to approximate the three aforementioned maps. We demonstrate through several simulations that PNNs are capable of handling very accurately several challenging tasks, including the motion of a particle in the electromagnetic potential, the nonlinear Schrödinger equation, and pixel observations of the two-body problem.**

***Index Terms*— Darboux–Lie theorem, geometric learning, physics-informed neural networks, SympNets.**

I. Introduction

THE connection between dynamical systems and neural network models has been widely studied in the literature, see [1]–[5]. In general, neural networks can be considered as discrete dynamical systems with the basic dynamics at each step being a linear transformation followed by a componentwise nonlinear (activation) function. In [6], the neural ODE is introduced as a continuous-depth model instead of specifying a discrete sequence of hidden layers. Even before the introduction of neural ODEs, a series of models with similar architectures had already been proposed to learn the hidden dynamics of a dynamical system in [7]–[9]. Backward error analysis of the discovery of dynamics is established and enriched in [10], where the inverse modified differential equations are introduced to understand the true dynamical system that is learned when the time derivatives are approximated by numerical schemes. The methods earlier do not assume the specific form of the equation *a priori*, however, as the physical systems usually possess intrinsic prior properties, other approaches also take into account the prior information of the systems.

For many physical systems from classical mechanics, the governing equation can be expressed in terms of Hamilton's equation. We denote the d-by-d identity matrix by I_d, and let

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

which is an orthogonal, skew-symmetric real matrix, so that $J^{-1} = J^T = -J$. The canonical Hamiltonian system can be written as

$$\dot{y} = J^{-1}\nabla H(y) \tag{1}$$

where $y(t) \in \mathbb{R}^{2d}$, and $H : \mathbb{R}^{2d} \to \mathbb{R}$ is the Hamiltonian, typically representing the energy of the system. It is well known that the phase flow of a Hamiltonian system is symplectic. Based on that observation, several numerical schemes which preserve symplecticity have been proposed to solve the forward problem in [11]–[13]. In recent works [14]–[19], the primary focus has been to solve the inverse problem, i.e., identifying Hamiltonian systems from data, using structured neural networks. For example, HNNs [15] use a neural network $\tilde{H}$ to approximate the Hamiltonian H in (1), then learn $\tilde{H}$ by reformulating the loss function. Based on HNNs, other models were proposed to tackle problems in generative modeling [16], [19] and continuous control [20]. Another line of approach is to learn the phase flow of the system directly while encoding the physical prior as symplecticity in the flow. Recently, we introduced symplectic networks (SympNets) with theoretical guarantees that they can approximate arbitrary symplectic maps [21]. Other networks of this type are presented in [22] and [23].

In practice, requiring a dynamical system to be Hamiltonian could be too restrictive, as the system has to be described in canonical coordinates. In [24], Lagrangian neural networks (LNNs) were introduced, which allow the system to be expressed in Cartesian coordinates. In [25], the models of HNNs and LNNs were generalized to constrained Hamiltonian neural networks (CHNNs) and constrained LNNs (CLNNs), enabling them to learn constrained mechanical systems written in Cartesian coordinates. In other developments, autoencoder-based HNNs (AE-HNNs) [15] and Hamiltonian generative networks (HGNs) [19] were proposed to learn and predict the images of mechanical systems, which can be seen

Manuscript received December 5, 2020; revised July 30, 2021 and November 19, 2021; accepted January 22, 2022. This work was supported in part by the Department of Energy (DOE) PhILMs Project under Grant DE-SC0019453 and in part by the Air Force Office of Scientific Research (AFOSR) Multidisciplinary University Research Initiative (MURI) Project under Grant FA9550-20-1-0358. The work of Ioannis G. Kevrekidis was supported in part by DARPA and in part by the Army Research Office. *(Corresponding author: George Em Karniadakis.)*

Pengzhan Jin is with the School of Mathematical Sciences, Peking University, Beijing 100871, China (e-mail: jpz@math.pku.edu.cn).

Zhen Zhang and George Em Karniadakis are with the Division of Applied Mathematics, Brown University, Providence, RI 02912 USA (e-mail: zhen_zhang1@brown.edu; george_karniadakis@brown.edu).

Ioannis G. Kevrekidis is with the Department of Chemical and Biomolecular Engineering, Johns Hopkins University, Baltimore, MD 21211 USA (e-mail: yannisk@jhu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3148734.

Digital Object Identifier 10.1109/TNNLS.2022.3148734

as Hamiltonian systems on manifolds embedded in high-dimensional spaces. Theoretically, Hamiltonian systems on manifolds written in noncanonical coordinates are equivalent to an important class of dynamical systems, namely, the Poisson systems. To wit, the Poisson systems take the form of

$$\dot{y} = B(y)\nabla H(y)$$

where $y \in \mathbb{R}^n$, n is not necessarily an even number; H is the Hamiltonian of the Poisson system, and the matrix-valued function $B(y)$ plays the role of J^{-1} in (1), which induces a general Poisson bracket as defined in Section II. The Darboux–Lie theorem states that a Poisson system can be turned into a Hamiltonian system by a coordinate transformation. As a consequence, structure-preserving numerical schemes for Poisson systems are normally developed by finding the coordinate transformation manually, then applying symplectic integrators on the transformed systems, see [12], [26].

Inspired by the Darboux–Lie theorem, we propose a novel neural network architecture, the Poisson neural network (PNN), to learn the phase flow of an arbitrary Poisson system, specifically, PNNs can learn any unknown diffeomorphism of Hamiltonian systems. The coordinate transformation and the phase flow of the transformed Hamiltonian system are parameterized by structured neural networks with physical priors. In the general setting, we use invertible neural networks (INNs) [27], [28] to represent the coordinate transformation. If all the data reside on a submanifold with dimension $2d < n$, AEs can be applied to approximate the coordinate transformation from the coordinates in $\mathbb{R}^n$ to its local coordinates as an alternative choice. This strategy is similar to [29], which learns dynamics in the latent space discovered through an AE. Compared with LNNs, PNNs are able to work on a more general coordinate system. Moreover, INN-based PNNs are able to learn multiple trajectories of a Poisson system on the whole data space simultaneously, while AE-HNNs, HGNs, CHNNs, and CLNNs are only designed to work on low-dimensional submanifolds of $\mathbb{R}^n$. Furthermore, our work lays a solid theoretical background for all the aforementioned models, suggesting that they are learning a Poisson system explicitly or implicitly.

The main contribution of this work is the development of a framework that learns general Poisson systems with structure-preserving models. To the best of our knowledge, this is the first work that achieves this. Most of the current works focus on Hamiltonian systems or constrained Hamiltonian systems. In fact, we can easily transform Poisson systems to Hamiltonian systems via a coordinate transformation. Furthermore, constrained Hamiltonian systems can be rewritten as Poisson systems and then transformed into Hamiltonian systems. Therefore, PNNs can be used to solve a wide range of problems. It is worth noting that a good learning model for Poisson systems should be structure preserving, i.e., can preserve the Poisson structure; otherwise, it is meaningless.

Another intriguing property of PNNs is that they are not only capable of learning Poisson systems, but are able to approximate an unknotted trajectory of an arbitrary autonomous system. We present related theorems, which indicate the great expressivity of PNNs. We demonstrate through computational experiments that PNNs can be practically useful in terms of learning high-dimensional autonomous systems, long-time prediction, and frame interpolation. PNNs also enjoy all the advantages listed in [21] as they are implemented based on the SympNets in this work. However, PNNs as a high-level architecture can also employ other modern symplectic neural networks and INNs.

The rest of this article is organized as follows. Section II introduces some necessary notation, terminology, and fundamental theorems that will be used. The learning theory for PNNs is presented in Section III. Section IV presents the experimental results for several Poisson systems and autonomous systems. A summary is given in Section V. Supporting materials, including the detailed implementation of PNNs, are included in the Appendix.

II. Preliminaries

The material required for this work is based on the mathematical background of the Hamiltonian system and its noncanonical form, i.e., the Poisson system. We refer the readers to [12] for more details.

A. Hamiltonian and Poisson Systems

First, we formally present the definitions of the Hamiltonian system and the Poisson system. We assume that all the functions or maps involved in this article are as smooth as needed.

Definition 1: The canonical Hamiltonian system takes the form

$$\dot{y} = J^{-1}\nabla H(y) \tag{2}$$

where $H : U \to \mathbb{R}$ is the Hamiltonian typically representing the energy of the system defined on the open set $U \subset \mathbb{R}^{2d}$.

System (2) can also be written in a general form by introducing the Poisson bracket.

Definition 2: Let $U \subset \mathbb{R}^n$ be an open set. The Poisson bracket $\{\cdot,\cdot\} : C^\infty(U) \times C^\infty(U) \to C^\infty(U)$ is a binary operation satisfying the following.

1) Anticommutativity

$$\{F, G\} = -\{G, F\}$$

2) Bilinearity

$$\{aF + bG, H\} = a\{F, H\} + b\{G, H\}$$
$$\{H, aF + bG\} = a\{H, F\} + b\{H, G\}, \quad a, b \in \mathbb{R}$$

3) Leibniz's rule

$$\{FG, H\} = \{F, H\}G + F\{G, H\}$$

4) Jacobi identity

$$\{\{F, G\}, H\} + \{\{H, F\}, G\} + \{\{G, H\}, F\} = 0$$

for $F, G, H \in C^\infty(U)$.

Consider the bracket

$$\{F, G\} = \sum_{i=1}^{d}\left(\frac{\partial F}{\partial q_i}\frac{\partial G}{\partial p_i} - \frac{\partial F}{\partial p_i}\frac{\partial G}{\partial q_i}\right) = \nabla F(y)^T J^{-1}\nabla G(y) \tag{3}$$

where $y = (p, q) = (p_1, \ldots, p_d, q_1, \ldots, q_d) \in \mathbb{R}^{2d}$. One can check that (3) is indeed a Poisson bracket. Then, system (2) can be written as

$$\dot{y}_i = \{y_i, H\}, \quad i = 1, \ldots, 2d$$

where y_i in the bracket denotes the map $y \to y_i$ for $y = (y_1, \ldots, y_{2d})$ by a slight abuse of notation. Now, we extend the bracket (3) to a general form as

$$\begin{aligned}(\{F, G\}_B)(y) &= \sum_{i,j=1}^{n} \frac{\partial F(y)}{\partial y_i} b_{ij}(y) \frac{\partial G(y)}{\partial y_j} \\ &= \nabla F(y)^T B(y) \nabla G(y) \end{aligned} \tag{4}$$

where $B(y) = (b_{ij}(y))_{n\times n}$ is a smooth matrix-valued function. Note that here we do not require n to be an even number. As many crucial properties of Hamiltonian systems rely uniquely on the conditions 1)–4) in Definition 2, we naturally expect the bracket (4) to be a Poisson bracket.

Lemma 1: The bracket defined in (4) is anticommutative, bilinear, and satisfies Leibniz's rule and the Jacobi identity if and only if

$$b_{ij}(y) = -b_{ji}(y) \quad \text{for all } i, j$$

and for all i, j, k

$$\sum_{l=1}^{n} \left(\frac{\partial b_{ij}(y)}{\partial y_l} b_{lk}(y) + \frac{\partial b_{jk}(y)}{\partial y_l} b_{li}(y) + \frac{\partial b_{ki}(y)}{\partial y_l} b_{lj}(y) \right) = 0.$$

Lemma 1 provides verifiable equivalence conditions for (4) to become a Poisson bracket. Then, we can give the definition of the Poisson system, which is actually the generalized form of the Hamiltonian system.

Definition 3: If a matrix-valued function $B(y)$ satisfies Lemma 1, then (4) defines a Poisson bracket, and the corresponding differential system

$$\dot{y} = B(y) \nabla H(y)$$

is a Poisson system. Here, H is still called a Hamiltonian.

Up to now, the Hamiltonian system and the Poisson system have been unified as

$$\dot{y}_i = \{y_i, H\}_B, \quad i = 1, \ldots, n \tag{5}$$

for B satisfying Lemma 1, and the system becomes Hamiltonian when $B = J^{-1}$.

B. Symplectic Map and Poisson Map

The study of the phase flows of the Hamiltonian and Poisson systems focuses on the symplectic map and the Poisson map.

Definition 4: A transformation $\Phi : U \to \mathbb{R}^{2d}$ (where U is an open set in $\mathbb{R}^{2d}$) is called a symplectic map if its Jacobian matrix satisfies

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

Definition 5: A transformation $\Phi : U \to \mathbb{R}^n$ (where U is an open set in $\mathbb{R}^n$) is called a Poisson map with respect to the Poisson system (5) if its Jacobian matrix satisfies

$$\left(\frac{\partial \Phi}{\partial y} \right) B(y) \left(\frac{\partial \Phi}{\partial y} \right)^T = B(\Phi(y)).$$

In fact, the phase flow $\phi_t^H(y)$ of the Hamiltonian system is a symplectic map, while the phase flow $\phi_t^P(y)$ of the Poisson system is a Poisson map, i.e., $\phi_t^H(y)$ and $\phi_t^P(y)$ satisfy Definitions 4 and 5, respectively. Based on these facts, we naturally expect the numerical methods or learning models for Hamiltonian systems and Poisson systems to preserve the intrinsic properties that $\phi_t^H(y)$ and $\phi_t^P(y)$ possess. So far, the numerical techniques for Hamiltonian systems have been well developed [11]–[13]; however, research on the Poisson systems is ongoing due to its complexity.

C. Coordinate Changes and the Darboux–Lie Theorem

The main idea in studying Poisson systems is to find the connection to Hamiltonian systems, which are easier to deal with. In fact, a Poisson system expressed in arbitrary new coordinates is again a Poisson system; hence, we naturally tend to simplify a given Poisson structure as much as possible by coordinate transformation.

Theorem 1 (Darboux 1882, Lie 1888): Suppose that the matrix $B(y)$ defines a Poisson bracket and is of constant rank $n - q = 2d$ in a neighborhood of $y_0 \in \mathbb{R}^n$. Then, there exist functions $P_1(y), \ldots, P_d(y)$, $Q_1(y), \ldots, Q_d(y)$, and $C_1(y), \ldots, C_q(y)$ satisfying

$$\begin{array}{lll} \{P_i, P_j\} = 0 & \{P_i, Q_j\} = -\delta_{ij} & \{P_i, C_l\} = 0 \\ \{Q_i, P_j\} = \delta_{ij} & \{Q_i, Q_j\} = 0 & \{Q_i, C_l\} = 0 \\ \{C_k, P_j\} = 0 & \{C_k, Q_j\} = 0 & \{C_k, C_l\} = 0 \end{array}$$

on a neighborhood of y_0, where δ_{ij} equals to 1 if $i = j$ else 0. The gradients of P_i, Q_i, and C_k are linearly independent, so that $y \to (P_i(y), Q_i(y), C_k(y))$ constitutes a local change of coordinates to canonical form.

Corollary 1 (Transformation to Canonical Form): Let us denote the transformation of Theorem 1 by $z = \theta(y) = (P_i(y), Q_i(y), C_k(y))$. With this change of coordinates, the Poisson system $\dot{y} = B(y)\nabla H(y)$ becomes

$$\dot{z} = B_0 \nabla K(z) \quad \text{with} \quad B_0 = \begin{pmatrix} J^{-1} & 0 \\ 0 & 0 \end{pmatrix}$$

where $K(z) = H(y)$. Writing $z = (p, q, c)$, this system becomes

$$\dot{p} = -K_q(p, q, c), \quad \dot{q} = K_p(p, q, c), \quad \dot{c} = 0.$$

Corollary 1 reveals the connection between Poisson systems and Hamiltonian systems via coordinate changes. In a forward problem, i.e., solving the Poisson system by numerical integration, transformations are available for many well-known systems to perform structure-preserving calculations [12], [26], but there does not exist a general method to search for the new coordinates of an arbitrary Poisson system, which is still an open research issue. However, the inverse problem, i.e., learning an unknown Poisson system based on data, is an easier task.

III. Learning Theory for Poisson Systems and Trajectories of Autonomous Systems

Assume that there is a dataset $\mathcal{T} = \{(x_i, y_i)\}_1^N$ $(x_i, y_i \in \mathbb{R}^n)$ from an unknown autonomous dynamical system that could be

a Poisson system or not, satisfying $\phi_h(x_i) = y_i$ for time step h and phase flow ϕ_t. We aim to discover the dynamics using learning models so that we can make predictions in the future or perform some other computational tasks. To describe things clearly, we first give the definition of the extended symplectic map.

Definition 6: A transformation $\Phi : U \to \mathbb{R}^n$ (where U is an open set in $\mathbb{R}^n$) is called an extended symplectic map with latent dimension $2d$ if it can be written as

$$\Phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \phi(x_1, x_2) \\ x_2 \end{pmatrix}, \quad x_1 \in \mathbb{R}^{2d}, \quad x_2 \in \mathbb{R}^{n-2d}, \quad 2d \le n$$

where ϕ is differentiable, and $\phi(\cdot, x_2)$ is a symplectic map for each fixed x_2. Note that Φ degenerates to a general symplectic map when $2d = n$.

A. Poisson Neural Networks

We propose a high-level network architecture, i.e., the PNNs, to learn Poisson systems or autonomous flows based on the Darboux–Lie theorem. Theorem 1 and Corollary 1 indicate that any Poisson system in n-dimensional space can be transformed to a "piecewise" Hamiltonian system, where $2d \le n$ is the latent dimension determined by the rank of $B(y)$. The architecture is composed of three parts: 1) a coordinate transformation; 2) an extended symplectic map; and 3) the inverse of the transformation, denoted by θ, Φ, and θ^{-1}, respectively. For the construction of extended symplectic neural networks (E-SympNets), we refer the readers to Appendix C-A, which is an important part of this work. The transformations θ and θ^{-1} can be implemented using two alternative approaches.

1) Primary Architecture: We model θ as an INN, as in [27] and [28], to automatically obtain its inverse θ^{-1}. Then, we learn the data by optimizing the mean-squared-error loss

$$L(\mathcal{T}) = \frac{1}{n \cdot N} \sum_{i=1}^{N} \left\| \theta^{-1} \circ \Phi \circ \theta(x_i) - y_i \right\|^2$$

where $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ is an E-SympNet with latent dimension $2d$.

2) Alternative Architecture: We exploit an AE to parameterize θ, θ^{-1} with two different neural networks. Then, the loss is designed as

$$\begin{aligned} &L(\mathcal{T}) \\ &= L_s(\mathcal{T}) + \lambda \cdot L_a(\mathcal{T}) \\ &= \frac{1}{2d \cdot N} \sum_{i=1}^{N} \|\Phi \circ \theta(x_i) - \theta(y_i)\|^2 \\ &\quad + \lambda \cdot \frac{1}{n \cdot N} \sum_{i=1}^{N} \left(\left\| \theta^{-1} \circ \theta(x_i) - x_i \right\|^2 + \left\| \theta^{-1} \circ \theta(y_i) - y_i \right\|^2 \right) \end{aligned}$$

where $\Phi : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ is a symplectic neural network and λ is a hyperparameter to be tuned. Note that in this case, $\theta^{-1} \circ \theta$ is not intrinsically equivalent to the identity map. Basically, this architecture only learns the Poisson map limited on a submanifold embedded in the whole phase space.

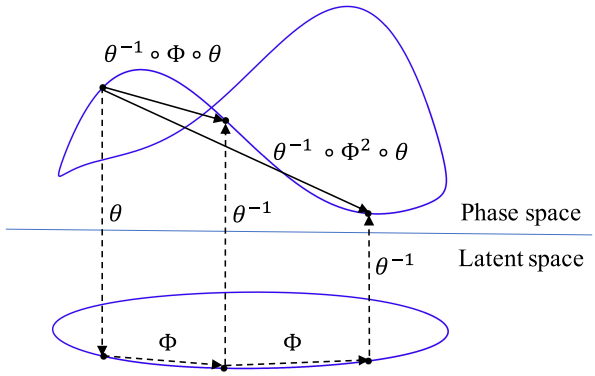


Fig. 1. Architecture of PNNs. PNNs are composed of three parts: 1) a coordinate transformation; 2) an extended symplectic map; and 3) the inverse of the transformation, denoted by θ, Φ, and θ^{-1}, respectively. θ and θ^{-1} are implemented by INNs for the primary architecture and by AE for the alternative architecture, while Φ is implemented by the E-SympNet, see Appendix C for details. The dimension of the latent space is equal to the original phase space, and the key difference is that the transformed system in latent space is "piecewise" Hamiltonian of latent dimension $2d$, while the whole space is of dimension $n \ge 2d$.

In both cases, we perform predictions by $f^k_{\text{PNN}} = \theta^{-1} \circ \Phi^k \circ \theta$, which gives the kth step. We prefer the INNs because the reconstruction loss will disappear compared with the AE, and we also expect to impose further prior information on the transformation. For example, in Section IV-C, we adopt a volume-preserving network as the INN, the so-called volume-preserving PNN (VP-PNN), which achieves better generalization compared with the nonvolume-preserving PNN (NVP-PNN), since the original Poisson system has a volume-preserving phase flow. More crucially, AE-based PNNs are unable to learn data lying on the whole space, rather than a $2d$-dimensional submanifold, when $2d < n$. In particular, the coordinate transformation twists the whole space and flatten a series of $2d$-dimensional submanifolds to planes, while the AE only flattens a single $2d$-dimensional submanifold since it loses many degrees of freedom upon dimension reduction. However, AE-based PNNs can perform better in some situations, such as in the numerical case in Section IV-E. Intuitively, the alternative architecture outperforms the primary architecture when $2d \ll n$. An illustration of PNNs is presented in Fig. 1.

B. Learning Poisson Systems

Consider the case that $\mathcal{T}$ consists of data points from a Poisson system. Next, we present the approximation properties of PNNs.

Theorem 2: Suppose that: 1) the E-SympNets Φ are universal approximators within the space of continuously differentiable extended symplectic maps in C^1 topology; 2) (primary architecture) the INNs θ are universal approximators within the space of invertible continuously differentiable maps which have nondegenerate Jacobian in the C^1 topology; and 3) (alternative architecture) the transformation neural networks θ and θ^{-1} are universal approximators within the space of continuous maps in the C^0 topology. Then, the corresponding PNNs $\theta^{-1} \circ \Phi \circ \theta$ are able to approximate arbitrary Poisson maps in C^1 topology for the primary architecture and are able to

approximate arbitrary Poisson maps (limited on submanifolds) within the space of continuous maps in C^0 topology for the alternative architecture. The approximations are considered on compact sets.

Proof: It can be deduced directly from Theorem 1 and Corollary 1. □

Notice that in the alternative architecture, the PNN $\theta^{-1}\circ\Phi\circ\theta$ itself is not intrinsically a Poisson map, however, by using $f_{\text{PNN}}^k = \theta^{-1} \circ \Phi^k \circ \theta$ for multistep prediction, it can also preserve the geometric structure and enjoy stable long-term performance in practice.

C. Learning Trajectories of Autonomous Systems

Now consider the case that $\mathcal{T}$ consists of a series of data points on a single trajectory (maybe not from a Poisson system), i.e., $\mathcal{T} = \{(x_{i-1}, x_i)\}_1^N$, where $\phi_h(x_{i-1}) = x_i$ for time step h and ϕ_t, which is the phase flow of an unknown autonomous system $\dot{y} = f(y)$. Unlike the theory for learning Poisson systems, the use of PNNs to learn autonomous flows is quite novel. The theories are driven by the observation that symplectic neural networks can learn a trajectory from a non-Hamiltonian system, see Section IV-A for details. The next theorem reveals the internal mechanism.

Theorem 3: Suppose that $U \subset \mathbb{R}^2$ is a simply connected open set, and the periodic solution $y(t) \in U$ is from an autonomous dynamical system

$$\dot{y} = f(y), \quad y \in U$$

then there exists a Hamiltonian $H(y)$, such that $y(t)$ also satisfies the Hamiltonian system

$$\dot{y} = J^{-1}\nabla H(y), \quad y \in U.$$

Proof: The proof can be found in Appendix B-A. □

Based on the above-mentioned theorem, one may be able to apply symplectic neural networks to arbitrary periodic solutions to the autonomous system in $\mathbb{R}^2$. Naturally, we tend to explore similar results in high-dimensional space. We intuitively expect to transform any high-dimensional periodic solution to an autonomous system into one lying on a plane with the help of coordinate changes, and subsequently, the original trajectory can be learned via PNN. In fact, this conjecture is almost right, except for the case when the orbit of the considered motion is a nontrivial 1-knot in $\mathbb{R}^3$. For the theory on knots, we refer to [30]–[32], and we briefly present the basic concepts in Appendix A.

Theorem 4: Suppose that $U \subset \mathbb{R}^n$ is a contractible open set, periodic solution $y(t) \in U$ is from an autonomous dynamical system

$$\dot{y} = f(y), \quad y \in U$$

and the orbit of $y(t)$ is unknotted. Then, there exists a Hamiltonian $H(y)$ and a $B(y)$ satisfying Lemma 1 with rank of 2, such that $y(t)$ also satisfies the Poisson system

$$\dot{y} = B(y)\nabla H(y), \quad y \in U.$$

Note that nontrivial 1-knots exist only in $\mathbb{R}^3$.

Proof: The proof can be found in Appendix B-B. □

Up to now, we have shown that PNNs can be used to learn almost any periodic solution to autonomous systems, and the latent dimension is actually fixed as 2. The symplectic structure embedded in PNNs will endow the predictions with long-term stability and more accuracy, for periodic solutions. Nevertheless, there is still a limitation of this method, as one can see, PNNs are allowed to learn only a single trajectory upon training. Basically, the limitation is inevitable as we have already got rid of most of the constraints on the vector field f, which is in fact a trade-off between data and systems. In spite of this fact, we still expect to further develop a strategy to relax the requirements of "single" and "periodic," by increasing the latent dimension. We conjecture it as follows.

Suppose that $U \subset \mathbb{R}^n$ is a contractible open set, $f : U \to \mathbb{R}^n$ is a vector field, and S is a smooth trivial $(2d-1)$-knot embedded in U. If $f|_S$ is a smooth tangent vector field on S, then there exists a smooth single-valued function $H(y)$ and a smooth matrix-valued function $B(y)$ satisfying Lemma 1 with rank of $2d$, such that $B(y)\nabla H(y)|_S = f|_S$.

The conjecture provides a more general insight into the above-mentioned theoretical results on a single trajectory. If it holds, one may learn several trajectories lying on a higher dimensional trivial knot simultaneously upon training, with a higher latent dimension. Unfortunately, the proofs of the 1-knot case cannot be easily extended to the general case, since the fact that a solenoidal vector field on $\mathbb{R}^2$ is exactly a field of the Hamiltonian system does not hold for higher dimensional space. A more thorough explanation of this conjecture is needed in future works.

IV. Simulation Results

In this section, we present several simulation cases to verify our theoretical results and indicate the potential application of PNNs in the field of computer vision. All the hyperparameters for detailed architectures and training settings are shown in Appendix C and Table III. For each detailed Poisson system involved, we obtain the ground truth and data using a high-order symplectic integrator with its corresponding coordinate transformation, which is listed in Appendix D. The codes are published in GitHub (https://github.com/jpzxshi/pnn).

A. Lotka–Volterra Equation

The Lotka–Volterra equation can be written as

$$\begin{pmatrix}\dot{u}\\ \dot{v}\end{pmatrix} = \begin{pmatrix}u(v-2)\\ v(1-u)\end{pmatrix} = \begin{pmatrix}0 & uv\\ -uv & 0\end{pmatrix}\nabla H(u,v) \tag{6}$$

where $H(u,v) = u - \ln u + v - 2\ln v$. As a Poisson system, we are able to discover the underlying symplectic structure using PNNs. The data consist of three trajectories, starting at $(1, 0.8)$, $(1, 1)$, and $(1, 1.2)$, respectively. We generate 100 training points with a time step $h = 0.1$ for each trajectory. Besides the PNN, we also use a SympNet to learn the three trajectories simultaneously, and the single trajectory starting at $(1, 1)$. We perform predictions for 1000 steps starting at the endpoints of the training trajectories, and the results of the three cases are presented in Fig. 2. As shown in the left-hand side of Fig. 2, the PNN successfully learns the system and

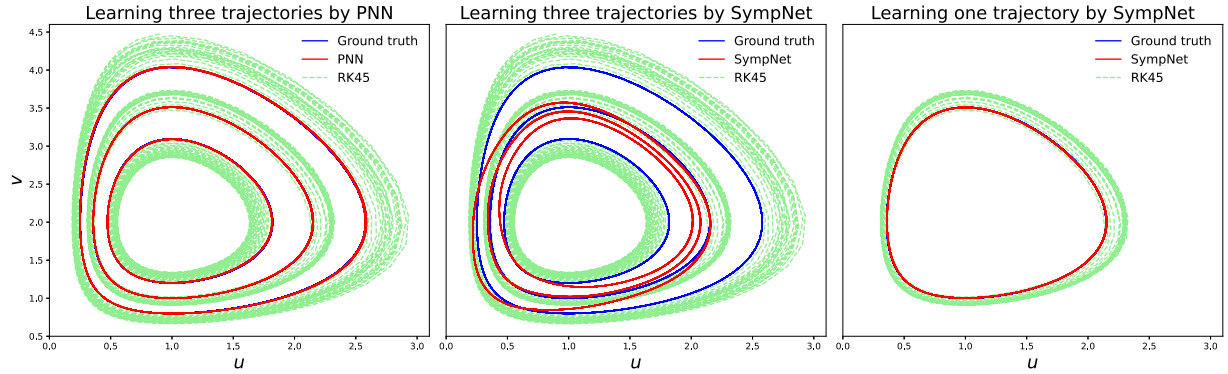


Fig. 2. Lotka–Volterra equation. Left: Learning three trajectories by PNN. The PNN successfully learns the system and achieves a stable long-time prediction. We also compute the trajectories by the classical Runge–Kutta method of order four (RK45) to show that a general numerical method cannot guarantee stability. Middle: Learning three trajectories by a SympNet. The SympNet fails to fit the three trajectories simultaneously since the learned system is not Hamiltonian. Right: Learning a single trajectory by a SympNet. The SympNet is able to learn this single trajectory due to Theorem 3.

achieves a stable long-time prediction. We also compute the trajectories by the classical Runge–Kutta method of order four (RK45) to show that solving the given system via a general numerical method cannot guarantee stability. Meanwhile, the SympNet [21] fails to fit the three trajectories simultaneously, since the data points are not from a Hamiltonian system, as shown in the middle of Fig. 2. However, the right-hand side of Fig. 2 reveals that the SympNet is indeed able to learn a single trajectory, even though it is not from a Hamiltonian system, which is consistent with Theorem 3.

B. Extended Pendulum System

We test the performance of a PNN on odd-dimensional Poisson systems. The motion of the pendulum system is governed by

$$\begin{pmatrix}\dot{p}\\ \dot{q}\end{pmatrix}=\begin{pmatrix}-\sin q\\ p\end{pmatrix}=\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\nabla H(p,q)$$

where $H(p,q)=(1/2)p^2-\cos q$. This is a canonical Hamiltonian system, and we subsequently extend this system to 3-D space

$$\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{c}\end{pmatrix}=\begin{pmatrix}-\sin q\\ p+c\\ 0\end{pmatrix}=\begin{pmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}\nabla\tilde{H}(p,q,c)$$

where $\tilde{H}(p,q,c)=H(p,q)+pc$. To make the data more difficult to learn, a nonlinear transformation $(u,v,r)=\theta^{-1}(p,q,c)=(p,q,p^2+q^2+c)$ is applied to the extended phase space. The governing equation for the transformed system is as follows:

$$\begin{aligned}\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{r}\end{pmatrix}&=\begin{pmatrix}0 & -1 & -2v\\ 1 & 0 & 2u\\ 2v & -2u & 0\end{pmatrix}\begin{pmatrix}u-3u^2-v^2+r\\ \sin v-2uv\\ u\end{pmatrix}\\&=:B(u,v,r)\nabla K(u,v,r)\end{aligned}\tag{7}$$

where $K(u,v,r)=(1/2)u^2-\cos v+ur-u^3-uv^2$. One may readily verify that B satisfies Lemma 1; therefore, (7) is a Poisson system.

Three trajectories are simulated with initial conditions $x_0=(0,1,1^2)$, $(0,1.5,1.5^2+0.1)$, $(0,2,2^2+0.2)$, and time step $h=0.1$. We use data points obtained in the first 100 steps as our training set. Then, we perform predictions for 1000 steps starting at the endpoints of the training set; the results are shown in Fig. 3. From the left-hand side of Fig. 3, it can be seen that the predictions made by the PNN match the ground truth perfectly, remaining on the true trajectories after long times. Meanwhile, the PNN is able to recover the underlying structure of the system, as shown on the right-hand side. The trajectories of the system in the latent space are recovered as trajectories on parallel planes, which match the fact that the trajectories are generated from several different 2-D symplectic submanifolds.

C. Charged Particle in an Electromagnetic Potential

We consider the dynamics of the charged particle in an electromagnetic potential governed by the Lorentz force

$$m\ddot{x}=q(E+\dot{x}\times B)$$

where m is the mass, $x\in\mathbb{R}^3$ denotes the particle's position, q is the electric charge, $B=\nabla\times A$ denotes the magnetic field, and $E=-\nabla\varphi$ is the electric field with A,φ being the potentials. Let $\dot{x}=v$ be the velocity of the charged particle, then the governing equations of the particle's motion can be expressed as

$$\begin{aligned}\begin{pmatrix}\dot{v}\\ \dot{x}\end{pmatrix}&=\begin{pmatrix}-\frac{q}{m^2}\hat{B}(x) & -\frac{1}{m}I\\ \frac{1}{m}I & 0\end{pmatrix}\nabla H(v,x)\\H(v,x)&=\frac{1}{2}mv^Tv+q\varphi(x)\end{aligned}\tag{8}$$

where

$$\hat{B}(x)=\begin{pmatrix}0 & -B_3(x) & B_2(x)\\ B_3(x) & 0 & -B_1(x)\\ -B_2(x) & B_1(x) & 0\end{pmatrix}$$

for $B(x)=(B_1(x),B_2(x),B_3(x))$. Here, we test the dynamics with $m=1$, $q=1$, and

$$A(x)=\frac{1}{3}\sqrt{x_1^2+x_2^2}\cdot(-x_2,x_1,0),\quad\varphi(x)=\frac{1}{100\sqrt{x_1^2+x_2^2}}$$

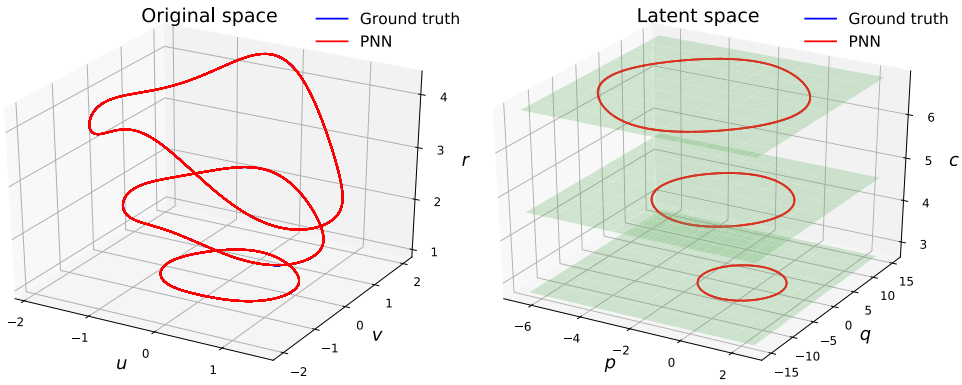


Fig. 3. Extended pendulum system. Left: Three ground truth trajectories compared with the predictions made by the PNN. The predictions match the ground truth perfectly without deviation. Right: The ground truth and the predictions expressed in the latent space by trained θ. The trajectories are lying on three different parallel planes.

for $x = (x_1, x_2, x_3)^T$. Then

$$B(x) = (\nabla \times A)(x) = \left(0, 0, \sqrt{x_1^2 + x_2^2}\right)$$
$$E(x) = -(\nabla\varphi)(x) = \frac{(x_1, x_2, 0)}{100\left(x_1^2 + x_2^2\right)^{\frac{3}{2}}}.$$

The initial state is chosen to be $v = (1, 0.5, 0)$ and $x = (0.5, 1, 0)$, in which case the system degenerates into 4-D dynamics, i.e., the motion of the particle is always on a plane. For simplicity, we also denote them by $v = (1, 0.5)$ and $x = (0.5, 1)$, and study the dimension-reduced system. We then generate a trajectory of 1500 training points followed by 300 test points with a time step $h = 0.1$. Subsequently, a volume-preserving PNN (VP-PNN) is trained to learn the training set, and we perform predictions for 2000 steps starting at the endpoint of the training set, as shown in Fig. 4. It can be seen that the VP-PNN perfectly predicts the trajectory without deviation. Furthermore, we also train a nonvolume-preserving PNN (NVP-PNN) and a volume-preserving neural network (VPNN) to compare with the above-mentioned model. After sufficient training, the three models make predictions starting at the initial state to reconstruct trajectories, as shown in Fig. 5. As one can see, the VP-PNN performs slightly better than the NVP-PNN, while the NVP-PNN is much better than the VPNN. The quantitative results shown in Table I also support this observation. Although the VP-PNN has larger training MSE and one step test MSE than the NVP-PNN, its long-time test MSE is instead less, which is not surprising because NVP-PNNs and VPNNs possess the prior information of Poisson structure and volume preservation respectively, while VP-PNNs has both of them. Note that the considered dimension-reduced system of (8) is source-free; hence, its phase flow is intrinsically volume-preserving on the 4-D space.

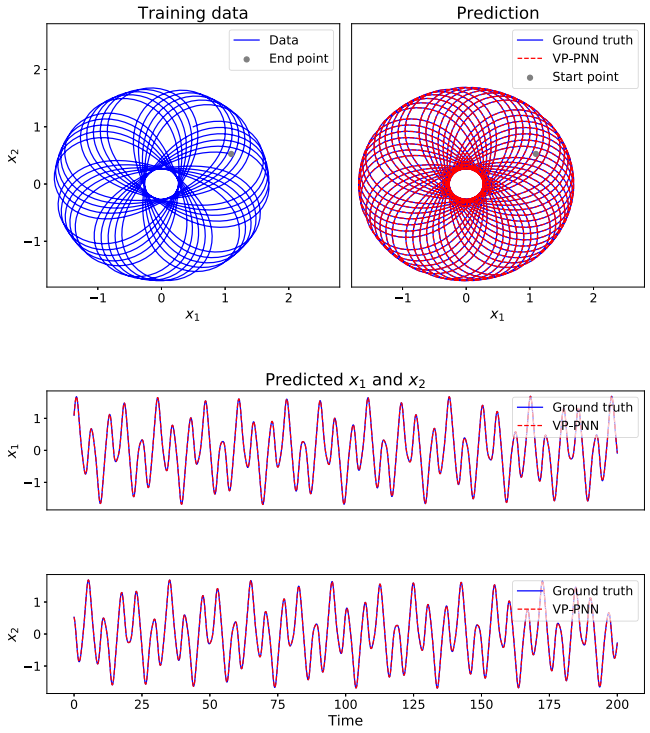


Fig. 4. Charged particle in the electromagnetic potential. Top-left: The position (x_1, x_2) of training flow. Top-right: Prediction of the VP-PNN starting at the endpoint of training flow for 2000 steps. The VP-PNN perfectly predicts the trajectory without deviation. Bottom: Prediction of the position of particle over time.

TABLE I

LOSSES OF THE THREE TRAINED MODELS. THE LONG-TIME TEST MSE IS EVALUATED ALONG 1000 STEPS STARTING AT THE ENDPOINT OF TRAINING DATA. ALTHOUGH VP-PNN HAS LARGER TRAINING MSE AND ONE STEP TEST MSE THAN NVP-PNN, ITS LONG-TIME TEST MSE IS INSTEAD LESS. THE VPNN PERFORMS WORSE THAN ABOVE TWO MODELS AND FAILS FOR LONG-TIME PREDICTION

	VP-PNN	NVP-PNN	VPNN
Training MSE	4.6×10^{-9}	1.6×10^{-9}	2.7×10^{-8}
Test MSE (One step)	1.5×10^{-8}	9.8×10^{-9}	6.1×10^{-8}
Test MSE (Long time)	1.1×10^{-5}	5.1×10^{-4}	8.8×10^{3}

D. Nonlinear Schrödinger Equation

We consider the nonlinear Schrödinger equation

$$\begin{cases} i\dfrac{\partial w}{\partial t} + \dfrac{\partial^2 w}{\partial x^2} + 2|w|^2 w = 0 \\ w(x, 0) = w_0(x) \end{cases}$$

where $w(x, t) = u(x, t) + iv(x, t)$ is a complex field and the boundary condition $w_0(x)$ is periodic, i.e., $w_0(x+1) = w_0(x)$ for $x \in \mathbb{R}$. An interesting space discretization of the nonlinear

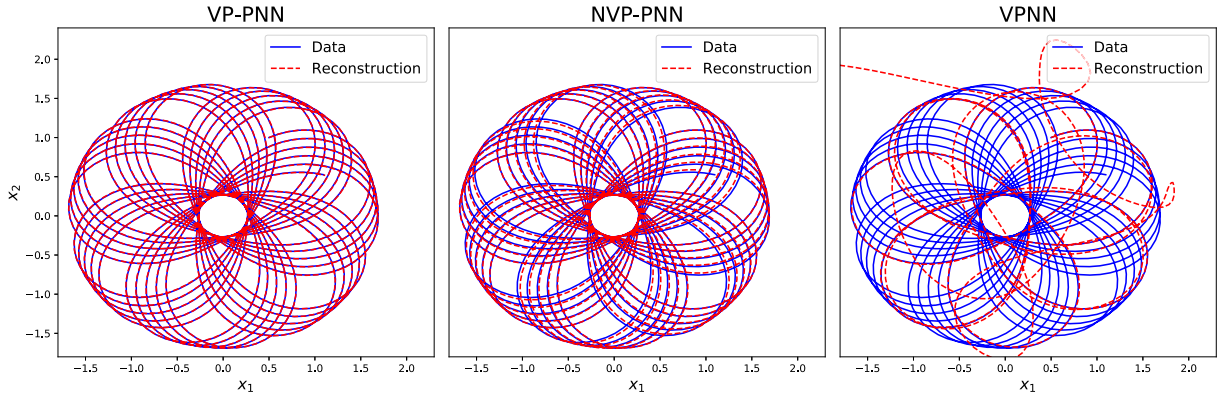


Fig. 5. Comparison of the reconstructed trajectories by three models. Left: The reconstructed trajectory by the VP-PNN, which perfectly matches the ground truth. Middle: The reconstructed trajectory by the NVP-PNN, which performs well, but slightly worse than the VP-PNN. Right: The reconstructed trajectory by the general VPNN, which fails to make long-time prediction.

Schrödinger equation is the Ablowitz–Ladik model

$$i\dot{w}_k + \frac{1}{\Delta x^2}(w_{k+1} - 2w_k + w_{k-1}) + |w_k^2|(w_{k+1} + w_{k-1}) = 0$$

with $w_k = w(k\Delta x, t)$, $\Delta x = 1/N$. Letting $w_k = u_k + iv_k$, we obtain

$$\dot{u_k} = -\frac{1}{\Delta x^2}(v_{k+1} - 2v_k + v_{k-1}) - \left(u_k^2 + v_k^2\right)(v_{k+1} + v_{k-1})$$

$$\dot{v_k} = \frac{1}{\Delta x^2}(u_{k+1} - 2u_k + u_{k-1}) + \left(u_k^2 + v_k^2\right)(u_{k+1} + u_{k-1}).$$

With $u = (u_1, \ldots, u_N)$, $v = (v_1, \ldots, v_N)$ this system can be written as

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & -D(u,v) \\ D(u,v) & 0 \end{pmatrix} \nabla H(u,v) \tag{9}$$

where $D = \operatorname{diag}(d_1, \ldots, d_N)$ is the diagonal matrix with entries $d_k(u,v) = 1 + \Delta x^2(u_k^2 + v_k^2)$, and the Hamiltonian is

$$H(u,v) = \frac{1}{\Delta x^2}\sum_{l=1}^{N}(u_l u_{l-1} + v_l v_{l-1}) - \frac{1}{\Delta x^4}\sum_{l=1}^{N}\ln\left(1 + \Delta x^2\left(u_l^2 + v_l^2\right)\right).$$

We, thus, get a Poisson system. In the experiment, we choose the boundary condition $u(x,0) = 2 + 0.2 \cdot \cos(2\pi x)$, $v(x,0) = 0$ and set $N = 20$; hence, (9) is a Poisson system of dimension 40. We then generate 500 training points followed by 100 test points with a time step $h = 0.01$. That means the solution to this equation during the time interval $[0, 5]$ is treated as the training set, and then, we learn the data using a PNN and predict the solution between $[5, 6]$. The result is shown in Fig. 6: both the real part and imaginary part match the ground truth well.

E. Pixel Observations of Two-Body Problem

We consider the pixel observations of the two-body problem, which are the images of two balls in black background, as shown in Fig. 7. Time series of the images form a movie of the motion of two balls governed by gravitation. Here, we intend to learn the phase flow on a coarse time grid

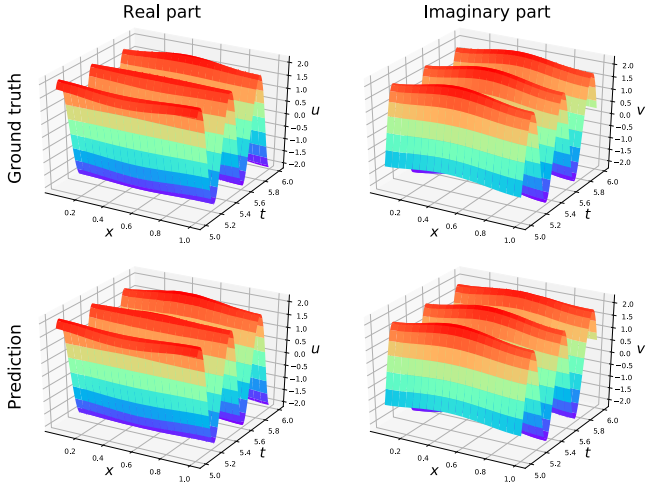


Fig. 6. Ablowitz–Ladik model of nonlinear Schrödinger equation. Predictions of both real and imaginary parts match the ground truth well.

while making predictions on a finer time grid, to forecast and smoothen the movie simultaneously. To achieve this goal, a simple recurrent training scheme is applied to our method. Similar treatments can be found in [33].

Suppose the training dataset is $\{x_n := x(n\Delta t)\}_{n=0}^{N}$. We set our goal to be making predictions on $x(t)$ at $t = (N + (n/m))\Delta t$, $n \geq 1$. Denote the PNN to be trained as $f_{\text{PNN}} = \theta^{-1} \circ \Phi \circ \theta$. Then, we train

$$f_{\text{PNN}}^m = \theta^{-1} \circ \Phi^m \circ \theta$$

to approximate the flow from x_i to x_{i+1}, and m is set to 2 in this case. Since we are learning a single trajectory on a submanifold of the high-dimensional space, AEs are used to approximate θ. λ in the corresponding loss function is chosen to be 1. After training, $f_{\text{PNN}}^k = \theta^{-1} \circ \Phi^k \circ \theta$ is used to generate predictions for $k \geq 1$.

A single trajectory $x(t)$ of the system is generated with time step $\Delta t = 0.6$, as shown in Fig. 7. The training dataset contains $N = 100$ images of size 100×50. Let $X_{\text{grid}} = (x_{N+1}, \ldots, x_{N+k})$ and $\tilde{X}_{\text{grid}} = (\tilde{x}_{N+1}, \ldots, \tilde{x}_{N+k})$ denote the ground truth and predictions made by the PNN at grid points. Let $X_{\text{mid}} = (x_{N+(1/2)}, \ldots, x_{N+k-(1/2)})$ and $\tilde{X}_{\text{mid}} = (\tilde{x}_{N+1}, \ldots, \tilde{x}_{N+k})$ denote the ground truth and predictions

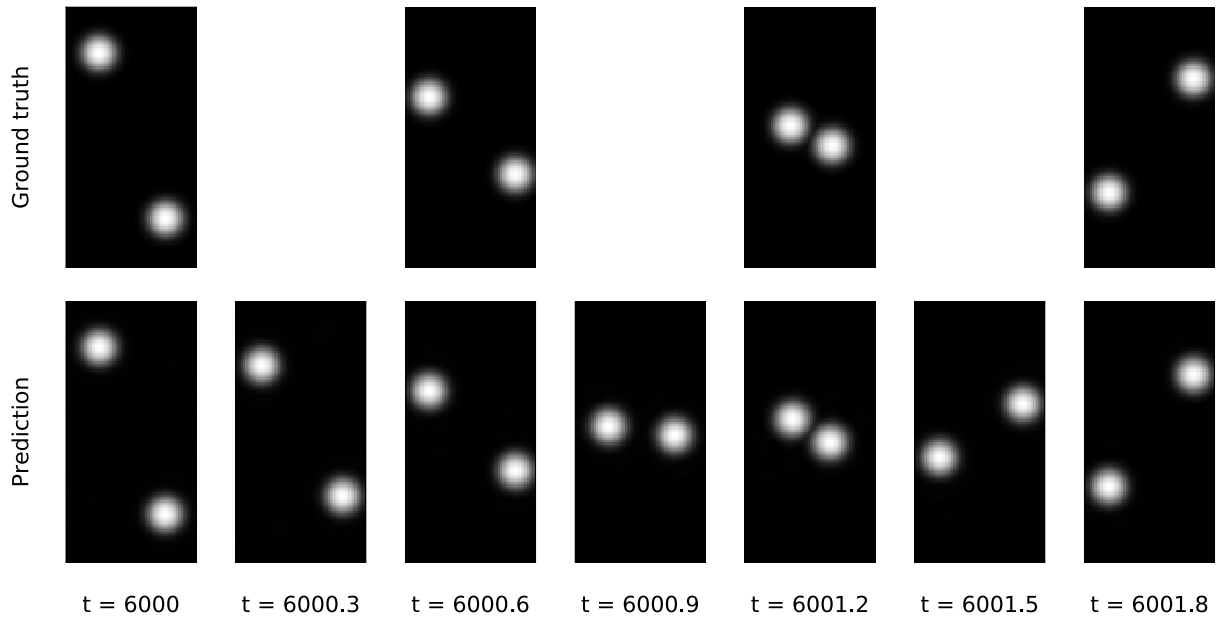


Fig. 7. Long-time prediction and frame interpolation for two body images. Top: Ground truth, four consecutive points in the test dataset, after $t = 6000$ with step size $h = 0.6$. Bottom: Predictions made by the PNN, with a finer step size $h = 0.3$. All: PNNs can handle long-time integration and frame interpolation perfectly.

made by the PNN on middle points of the grids. The test loss on grids is calculated as the mean squared error between X_{grid} and $\tilde{X}_{\text{grid}}$, while the test loss on middle points is calculated as the mean squared error between X_{mid} and $\tilde{X}_{\text{mid}}$. We use a similar definition as in [34] to compute the valid prediction time (VPT) T_ϵ. Suppose we are given the ground truth dataset x and prediction $\tilde{x}$, starting from $t = 0$. Let the root-mean-square error (RMSE) be

$$\mathcal{E}(\tilde{x}) = \sqrt{\langle (x - \tilde{x})^2 \rangle}$$

where $\langle \cdot \rangle$ stands for spatial average. The VPT is defined to be

$$T_\epsilon = \arg\max_{t_f} \{ t_f | \mathcal{E}(\tilde{x}(t)) \leq \epsilon \ \forall t \leq t_f \}$$

where ϵ is a hyperparameter to be chosen. Here, we set $\epsilon = 0.02$.

Low error is obtained both on the grid points and the middle points, as shown in Table II, which indicates that PNNs can handle prediction and interpolation simultaneously. The VPT is much longer than the time scale of the training window, further suggesting that PNNs are good at long-time predictions and intrinsically structure preserving. It can be seen in Fig. 7 that the prediction matches the ground truth perfectly even after $t = 6000$. We also compare the performance of AE-HNN and PNN in Fig. 8. The experimental details can be found in Appendix E. We find that PNNs significantly outperform AE-HNNs in terms of prediction accuracy due to their structure-preserving nature.

Note that according to Theorem 4, the periodic solution to an autonomous dynamical system in $\mathbb{R}^d$ can always be learned by PNNs when $d > 3$. Since this trajectory of two-body system is periodic in $\mathbb{R}^{100\times 50}$ and the image at step $n + 1$ is uniquely determined by the image at step n, we can assume without loss of generality that the pixel

TABLE II

LOSSES AND VPT OF THE PNN. THE TEST MSE IS EVALUATED ALONG 100 STEPS STARTING AT THE ENDPOINT OF DATA

Train MSE	Test MSE (Grid)	Test MSE (Middle)	VPT
1.7×10^{-6}	2.1×10^{-6}	2.0×10^{-6}	6308

observations of a two-body system form a periodic solution to an autonomous dynamical system, regardless of whether the internal mechanism is Hamiltonian.

V. CONCLUSION

The main contribution of this article is to provide a novel high-level network architecture, PNN, to learn and further extrapolate the phase flow of an arbitrary Poisson system while preserving its intrinsic Poisson structure. To the best of our knowledge, this is the first work that can achieve this. Since a single periodic solution to an autonomous system can be proven to be a solution to a Poisson system if the orbit is unknotted, PNNs can be directly applied to a much broader class of systems without modification. From this perspective, theoretical results regarding the approximation ability of PNNs are presented. We present several simulations including the Lotka-Volterra equation, an extended pendulum system, charged particles in the electromagnetic potential, a nonlinear Schrödinger equation, and a trajectory of the two-body image that support our theoretical findings and illustrate the advantages of PNNs on long-time prediction and frame interpolation. Even though not explicitly mentioned in this article, PNNs can be easily extended to learn phase flows from irregularly sampled data using the feature of SympNets. PNNs can also learn the Hamiltonian systems on low-dimensional submanifolds or constrained Hamiltonian systems, which can be expressed as a Poisson system. Note that PNN is very flexible as a high-level architecture, and one

may use other preferred symplectic neural networks or INNs as alternative approximators if needed.

Despite the great expressivity, stability, and interpretability of PNNs, an open issue is whether one can use them to infer and make predictions on multiple trajectories of an autonomous system without a generalized Poisson structure under certain circumstances. We conjectured in this article that the solution to an arbitrary autonomous system lying on a smooth trivial $(2d-1)$-knot matches the solution to a Poisson system. If this holds, the use of PNNs on multiple trajectories of a general autonomous system would be theoretically justified. We leave the proof or counterexamples of this conjecture as to future work.

Appendix A
Introduction to Knot

Consider the embeddings of S^n in S^m, $m \geq n+1$. Two embeddings k_1 and k_2 are equivalent if there is a homeomorphism h of S^m such that $h(k_1) = k_2$. An embedding $k : S^n \subset S^m$ is *unknotted* if it is equivalent to the trivial knot $k_0 : S^n \subset S^m$ defined by the standard embedding

$$k_0 : S^n \to S^m; \quad (x_0, \ldots, x_n) \to (x_0, \ldots, x_n, 0, \ldots, 0).$$

In fact, the embedding k is always unknotted when $m \neq n+2$. Therefore, an n-knot is defined as an embedding of S^n in S^{n+2}. For convenience, let us make a little change in the terminology: an n-knot means an embedding of S^n in S^m for $m \geq n+1$ since we are more concerned about the trivial knot in this work.

Appendix B
Proofs for Theorems

A. Proof of Theorem 3

In 2-D space, consider the system

$$\begin{cases} \Delta \boldsymbol{v} = \text{grad } p \\ \text{div } \boldsymbol{v} = 0 \end{cases}$$

with boundary condition

$$\boldsymbol{v}|_\Gamma = \boldsymbol{a}$$

where Γ is the orbit of $y(t)$ and $\boldsymbol{v} : U \to \mathbb{R}^2$ is a vector field. [35, p. 60, Th. 1] shows that there exists a solution $\boldsymbol{v}$ to the system earlier given a suitable single-valued function p and any continuous $\boldsymbol{a}$ satisfying

$$\oint_\Gamma \boldsymbol{a} \cdot \boldsymbol{n} ds = 0$$

where $\boldsymbol{n}$ is the exterior normal with respect to the domain inside Γ. Set $\boldsymbol{a} = f$, we have

$$\oint_\Gamma f \cdot \boldsymbol{n} ds = \oint_\Gamma 0 ds = 0$$

hence, there exists a $\boldsymbol{v}$ such that

$$\text{div } \boldsymbol{v} = 0, \quad \boldsymbol{v}|_\Gamma = f.$$

Note that $\boldsymbol{v}$ is a solenoidal vector field, which leads to

$$\boldsymbol{v} = \text{curl } (-H) = \left(-\frac{\partial H}{\partial y_2}, \frac{\partial H}{\partial y_1}\right)^T = J^{-1} \nabla H.$$

The H defined earlier is exactly the Hamiltonian we are looking for.

B. Proof of Theorem 4

Since $y(t)$ is unknotted, there exist a periodic $\gamma(t) = (\gamma_1(t), \gamma_2(t))^T \in \mathbb{R}^2$ and a homeomorphism $\theta : U \to U$ such that

$$\theta(y(t)) = \Gamma(t) := (\gamma_1(t), \gamma_2(t), 0, \ldots, 0)^T \in \mathbb{R}^n$$

which is exactly the coordinate transformation deforming $y(t)$ into $\Gamma(t)$. Theorem 3 shows that $\gamma(t)$ is also a solution to a Hamiltonian system; hence, the extended solution $\Gamma(t)$ satisfies

$$\begin{cases} \dot{\Gamma} = \begin{pmatrix} J^{-1} & 0 \\ 0 & 0 \end{pmatrix} \nabla K(\Gamma), \quad J \in \mathbb{R}^{2\times 2} \\ K(y_1, y_2, \ldots, y_n) = H(y_1, y_2) \end{cases} \tag{10}$$

for a single-valued function H. In fact, a Poisson system expressed in an arbitrary new coordinate immediately becomes a new Poisson system with a new $B(y)$ whose rank is equivalent to the original one [12, p. 265]. Therefore, $y(t)$ is also a solution to a Poisson system obtained by expressing system (10) in new coordinate via transformation θ. Furthermore, they share the same latent dimension of two.

Appendix C
Implementation of Architecture

A. Extended Symplectic Neural Networks

We adopt SympNets [21] as the universal approximators for symplectic maps. The architecture of SympNets is based on three modules given in the following.

1) *Linear Modules*

$$\mathcal{L}_n \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} I & 0/S_n \\ S_n/0 & I \end{pmatrix} \cdots \begin{pmatrix} I & 0 \\ S_2 & I \end{pmatrix} \begin{pmatrix} I & S_1 \\ 0 & I \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} + b$$
$$p, q \in \mathbb{R}^d$$

where $S_i \in \mathbb{R}^{d\times d}$ are symmetric, $b \in \mathbb{R}^{2d}$ is the bias, while the unit upper triangular symplectic matrices and the unit lower triangular symplectic matrices appear alternately. In this module, S_i (represented by $A_i + A_i^T$ in practice) and b are parameters to learn. In fact, $\mathcal{L}_n$ can represent any linear symplectic map [37].

2) *Activation Modules*

$$\mathcal{N}_{\text{up}} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p + \tilde{\sigma}_a(q) \\ q \end{pmatrix}$$
$$\mathcal{N}_{\text{low}} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p \\ \tilde{\sigma}_a(p) + q \end{pmatrix}, \quad p, q \in \mathbb{R}^d$$

where $\tilde{\sigma}_a(x) := a \odot \sigma(x)$ for $x \in \mathbb{R}^d$. Here, $\odot$ is the elementwise product, σ is the activation function, and $a \in \mathbb{R}^d$ is the parameter to learn.

3) *Gradient Modules*

$$\mathcal{G}_{\text{up}} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p + \hat{\sigma}_{K,a,b}(q) \\ q \end{pmatrix}$$
$$\mathcal{G}_{\text{low}} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} p \\ \hat{\sigma}_{K,a,b}(p) + q \end{pmatrix}, \quad p, q \in \mathbb{R}^d$$

where $\hat{\sigma}_{K,a,b}(x) := K^T(a \odot \sigma(Kx + b))$ for $x \in \mathbb{R}^d$. Here, $a, b \in \mathbb{R}^l$, $K \in \mathbb{R}^{l\times d}$ are the parameters to learn,

and l is a positive integer regarded as the width of the module.

The SympNets are the composition of the above-mentioned three modules. In particular, we use two classes of SympNets: the LA-SympNets composed of linear and activation modules and the G-SympNets composed of gradient modules. Notice that both LA and G SympNets are universal approximators for symplectic maps, as shown in [21]. For convenience, we clarify the terminology for describing a detailed LA(G)-SympNet: an LA-SympNet of m layers with n sublayers means it is the composition of m linear modules and $m-1$ activation modules, where linear and activation modules appear alternatively like the architecture of the fully connected neural network, and each linear module is composed of n alternated triangular symplectic matrices; a G-SympNet of m layers with width l means it is composed of m alternated gradient modules, and the width l is defined as earlier.

In this works, we further develop the E-SympNets by extending the gradient modules.

1) *Extended Modules*

$$\mathcal{E}_{\text{up}}\begin{pmatrix}p\\q\\c\end{pmatrix}=\begin{pmatrix}p+\widehat{\sigma}_{K_1,K_2,a,b}(q,c)\\q\\c\end{pmatrix}$$
$$\mathcal{E}_{\text{low}}\begin{pmatrix}p\\q\\c\end{pmatrix}=\begin{pmatrix}p\\\widehat{\sigma}_{K_1,K_2,a,b}(p,c)+q\\c\end{pmatrix}$$
$$p,q\in\mathbb{R}^d,\quad c\in\mathbb{R}^{n-2d}$$

where $\widehat{\sigma}_{K_1,K_2,a,b}(x,c):=K_1^T(a\odot\sigma(K_1x+K_2c+b))$ for $x\in\mathbb{R}^d$, $c\in\mathbb{R}^{n-2d}$. Here, $a,b\in\mathbb{R}^l$, $K_1\in\mathbb{R}^{l\times d}$, $K_2\in\mathbb{R}^{l\times(n-2d)}$ are the parameters to learn, and l is a positive integer regarded as the width of the module.

The extended symplectic neural networks (E-SympNets) are the composition of extended modules. As noticed, $2d$ is the latent dimension of the E-SympNets. The terminology for describing the architecture of E-SympNets is the same as that for G-SympNets; hence, we will not repeat it here. It is worth mentioning that the approximation property of E-SympNets is still unknown, which is left as future work.

B. Invertible Neural Networks

In numerical experiments, we adopt the non-linear independent components estimation (NICE) [27] as volume-preserving invertible neural networks (INNs) and the real NVP [28] as general nonvolume-preserving INNs. The approximation capabilities of NICE are analyzed in [38].

1) *Volume-Preserving Modules*

$$\mathcal{V}_{\text{up}}\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}x_1+m_1(x_2)\\x_2\end{pmatrix}$$
$$\mathcal{V}_{\text{low}}\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}x_1\\m_2(x_1)+x_2\end{pmatrix}$$
$$x_1\in\mathbb{R}^d,\quad x_2\in\mathbb{R}^{n-d}$$

where $m_1:\mathbb{R}^{n-d}\to\mathbb{R}^d$ and $m_2:\mathbb{R}^d\to\mathbb{R}^{n-d}$ are modeled as fully connected neural networks.

2) *Nonvolume-Preserving Modules*

$$\mathcal{C}_{\text{up}}\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}x_1\odot\exp s_1(x_2)+t_1(x_2)\\x_2\end{pmatrix}$$
$$\mathcal{C}_{\text{low}}\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{pmatrix}x_1\\x_2\odot\exp s_2(x_1)+t_2(x_1)\end{pmatrix}$$
$$x_1\in\mathbb{R}^d,\quad x_2\in\mathbb{R}^{n-d}$$

where $s_1,t_1:\mathbb{R}^{n-d}\to\mathbb{R}^d$ and $s_2,t_2:\mathbb{R}^d\to\mathbb{R}^{n-d}$ are modeled as fully connected neural networks.

Basically, the volume-preserving (nonvolume-preserving) INNs are the alternated composition of upper and lower volume-preserving (nonvolume-preserving) modules. We say a volume-preserving (nonvolume-preserving) INN is of m layers and n sublayers with width l if it is composed of m alternated volume-preserving (nonvolume-preserving) modules and each of m_1,m_2 (s_1,t_1,s_2,t_2) is a fully-connected neural network (FNN) of n layers with width l. Meanwhile, the partition dimension d is also fixed as an architecture parameter and, in practice, we intuitively set $d\approx n/2$ for more expressivity. Note that this d has nothing to do with the latent dimension of E-SympNets.

C. Autoencoder

We apply the traditional autoencoder (AE) [39] to the alternative architecture for dimension-reduced cases. Suppose that the latent dimension is $\text{rank}(B(y))=2d<n$, then the AE is composed of two fully connected neural networks, i.e., an encoder $f_e:\mathbb{R}^n\to\mathbb{R}^{2d}$ and a decoder $f_d:\mathbb{R}^{2d}\to\mathbb{R}^n$, whose hidden nodes are all not less than $2d$. Note that the FNNs can also be replaced by other useful architectures like CNNs if needed. In Section IV, the encoder and decoder are chosen of the same depth and width.

APPENDIX D
TRANSFORMATIONS FOR POISSON SYSTEMS

Here, we briefly present the transformations to Hamiltonian systems for the involved Poisson systems in this work.

A. Lotka–Volterra Equation

With coordinate transformation $(p,q)=(\ln u,\ln v)$, system (6) can be written as

$$\begin{pmatrix}\dot{p}\\\dot{q}\end{pmatrix}=J^{-1}\nabla H(p,q),\quad H(p,q)=p-\exp(p)+2q-\exp(q).$$

B. Charged Particle in the Electromagnetic Potential

By considering the position x and the conjugate momentum $p=m\dot{x}+qA(x)$, the Poisson system (8) can be written in the canonical form

$$\begin{pmatrix}\dot{p}\\\dot{x}\end{pmatrix}=J^{-1}\nabla H(p,x)$$
$$H(p,x)=\frac{1}{2m}(p-qA(x))^T(p-qA(x))+q\varphi(x).$$

TABLE III

MODEL ARCHITECTURE. THE DETAILED IMPLEMENTATIONS OF SYMPLECTIC NEURAL NETWORKS, INNS, THE AE, AND THEIR CORRESPONDING TERMINOLOGY USED IN THIS TABLE ARE SHOWN IN APPENDIX C. THE SIGMOID ACTIVATION FUNCTIONS ARE USED FOR ALL THE MODELS. THE OPTIMIZER IS CHOSEN TO BE ADAM [36] WITH LEARNING RATE 0.001. THE NUMBERS OF TRAINING ITERATIONS ARE 2×10^5, 1×10^5, 2×10^6, 1×10^6, AND 5×10^5, RESPECTIVELY, FOR THE FIVE PROBLEMS CONSIDERED

Problem		Section IV-A		Section IV-B	Section IV-C		Section IV-D	Section IV-E
		PNN	SympNet		PNN	VPNN		
θ	Type	NVP	-	NVP	VP/NVP	VP	NVP	AE
	Partition	1	-	2	2	2	20	-
	Layers	3	-	3	10	20	3	2
	Sublayers	2	-	2	3	3	2	-
	Width	30	-	30	50	50	100	50
Φ	Type	G	G	E	G	-	G	LA
	Layers	3	6	3	10	-	10	3
	Sublayers	-	-	-	-	-	-	2
	Width	30	30	30	50	-	100	-

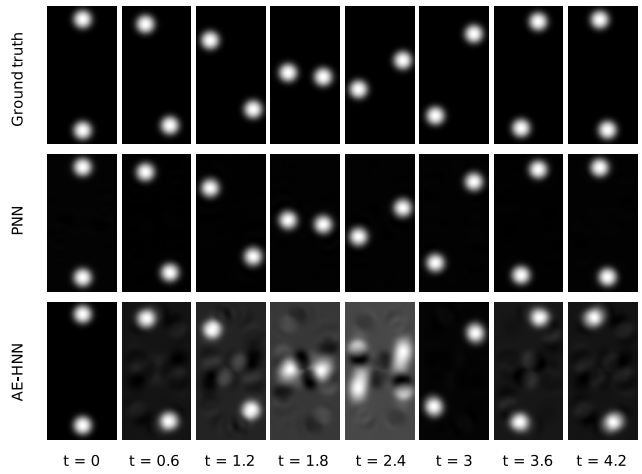


Fig. 8. Comparison between PNN and AE-HNN for two body images. Top: Ground truth, eight consecutive points in the dataset, after $t = 0$ with step size $h = 0.6$. Bottom: Predictions made by the AE-HNNs, which become blurry after a few time steps and do not capture the correct period of the system. Middle: Predictions made by the PNNs, which match the ground truth well.

C. Nonlinear Schrödinger Equation

A transformation has been proposed to make (9) a canonical Hamiltonian system

$$\begin{cases} p_k = u_k \sigma\left(\Delta x^2\left(u_k^2 + v_k^2\right)\right) \\ q_k = v_k \sigma\left(\Delta x^2\left(u_k^2 + v_k^2\right)\right), \end{cases} \quad \text{with} \quad \sigma(x) = \sqrt{\frac{\ln(1+x)}{x}}$$

which treats the variables symmetrically. Its inverse is

$$\begin{cases} u_k = p_k \tau\left(\Delta x^2\left(p_k^2 + q_k^2\right)\right) \\ v_k = q_k \tau\left(\Delta x^2\left(p_k^2 + q_k^2\right)\right), \end{cases} \quad \text{with} \quad \tau(x) = \frac{\exp x - 1}{x}.$$

Now, the Hamiltonian in the new variables is

$$H(p,q) = \frac{1}{\Delta x^2} \sum_{l=1}^{N} \tau\left(\Delta x^2\left(p_l^2 + q_l^2\right)\right) \tau\left(\Delta x^2\left(p_{l-1}^2 + q_{l-1}^2\right)\right) \cdot \left(p_l p_{l-1} + q_l q_{l-1}\right) - \frac{1}{\Delta x^2} \sum_{l=1}^{N} \left(p_l^2 + q_l^2\right).$$

APPENDIX E
COMPARISON WITH BASELINE

We compare the performance between PNN and AE-HNN [15] on the two-body image problem in the same setting as IV-E. The detailed architecture for PNN is already given in Table III. For AE-HNN, the layers and width of the AE are chosen to be 4 and 200; while the layers and width of HNN are chosen to be 3 and 200. The latent dimension is chosen to be 4 and the activation function is chosen to be hyperbolic tangent. We find that PNN significantly outperforms AE-HNN in a few time steps, as shown in Fig. 8. The predictions of AE-HNN become blurry and do not capture the correct period of the system.

REFERENCES

[1] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, "Reversible architectures for arbitrarily deep residual neural networks," 2017, *arXiv:1709.03698*.

[2] E. Weinan, "A proposal on machine learning via dynamical systems," *Commun. Math. Statist.*, vol. 5, no. 1, pp. 1–11, 2017.

[3] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, no. 1, Jan. 2018, Art. no. 014004.

[4] Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3276–3285.

[5] Y. Lu, C. Ma, Y. Lu, J. Lu, and L. Ying, "A mean-field analysis of deep ResNet and beyond: Towards provable optimization via overparameterization from depth," 2020, *arXiv:2003.05508*.

[6] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6571–6583.

[7] R. Rico-Martinez, J. S. Anderson, and I. G. Kevrekidis, "Continuous-time nonlinear signal processing: A neural network based approach for gray box identification," in *Proc. IEEE Workshop Neural Netw. Signal Process.*, Sep. 1994, pp. 596–605.

[8] R. González-García, R. Rico-Martínez, and I. Kevrekidis, "Identification of distributed parameter systems: A neural net based approach," *Comput. Chem. Eng.*, vol. 22, pp. S965–S968, Mar. 1998.

[9] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," 2018, *arXiv:1801.01236*.

[10] A. Zhu, P. Jin, B. Zhu, and Y. Tang, "Inverse modified differential equations for discovery of dynamics," 2020, *arXiv:2009.01058*.

[11] K. Feng, "On difference schemes and symplectic geometry," in *Proc. 5th Int. Symp. Differ. Geometry Differ. Equ.*, 1984, pp. 6–16.

[12] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, vol. 31. Berlin, Germany: Springer, 2006.

[13] C. Lubich, *From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis*. Zürich, Switzerland: European Mathematical Society, 2008.

[14] T. Bertalan, F. Dietrich, I. Mezić, and I. G. Kevrekidis, "On learning Hamiltonian systems from data," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 29, no. 12, Dec. 2019, Art. no. 121107.

[15] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 15353–15363.

[16] D. J. Rezende, S. Racanière, I. Higgins, and P. Toth, "Equivariant Hamiltonian flows," 2019, *arXiv:1909.13739*.

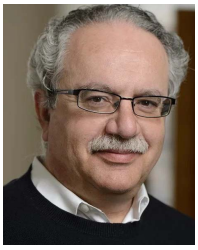
[17] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia, "Hamiltonian graph networks with ODE integrators," 2019, *arXiv:1909.12790*.
[18] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou, "Symplectic recurrent neural networks," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–23.
[19] P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins, "Hamiltonian generative networks," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–19.
[20] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ODE-Net: Learning Hamiltonian dynamics with control," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–16.
[21] P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis, "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems," *Neural Netw.*, vol. 132, pp. 166–179, Dec. 2020.
[22] D. DiPietro, S. Xiong, and B. Zhu, "Sparse symplectically integrated neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6074–6085.
[23] S. Xiong, Y. Tong, X. He, S. Yang, C. Yang, and B. Zhu, "Nonseparable symplectic neural networks," 2020, *arXiv:2010.12636*.
[24] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," 2020, *arXiv:2003.04630*.
[25] M. Finzi, K. A. Wang, and A. G. Wilson, "Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 13880–13889.
[26] Y. Tang, L. Vázquez, F. Zhang, and V. Pérez-García, "Symplectic methods for the nonlinear Schrödinger equation," *Comput. Math. With Appl.*, vol. 32, no. 5, pp. 73–83, 1996.
[27] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear independent components estimation," 2014, *arXiv:1410.8516*.
[28] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," 2016, *arXiv:1605.08803*.
[29] R. Rico-Martinez, K. Krischer, I. Kevrekidis, M. Kube, and J. Hudson, "Discrete-vs. continuous-time nonlinear signal processing of Cu electrodissolution data," *Chem. Eng. Commun.*, vol. 118, no. 1, pp. 25–48, 1992.
[30] C. Livingston, *Knot Theory*, vol. 24. Cambridge, U.K.: Cambridge Univ. Press, 1993.
[31] M. A. Armstrong, *Basic Topology*. New York, NY, USA: Springer, 2013.
[32] A. Ranicki, *High-Dimensional Knot Theory: Algebraic Surgery in Codimension 2*. Berlin, Germany: Springer, 2013.
[33] J. S. Anderson, I. G. Kevrekidis, and R. Rico-Martinez, "A comparison of recurrent training algorithms for time series analysis and system identification," *Comput. Chem. Eng.*, vol. 20, pp. S751–S756, Jan. 1996.
[34] P. Vlachas *et al.*, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," *Neural Netw.*, vol. 126, pp. 191–217, Jun. 2020.
[35] O. A. Ladyzhenskaya, *The Mathematical Theory of Viscous Incompressible Flow*, vol. 2. New York, NY, USA: Gordon and Breach, 1969, pp. 1–224.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.
[37] P. Jin, Y. Tang, and A. Zhu, "Unit triangular factorization of the matrix symplectic group," *SIAM J. Matrix Anal. Appl.*, vol. 41, no. 4, pp. 1630–1650, Jan. 2020.
[38] A. Zhu, P. Jin, and Y. Tang, "Approximation capabilities of measure-preserving neural networks," 2021, *arXiv:2106.10911*.
[39] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J.*, vol. 37, no. 2, pp. 233–243, Feb. 1991.

Pengzhan Jin received the Ph.D. degree in computational mathematics from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China, in 2021.

He is currently a Post-Doctoral Researcher with the School of Mathematical Sciences, Peking University, Beijing. His research interest includes intersection of machine learning and scientific computing.

Zhen Zhang received the bachelor's degree in computing mathematics from the City University of Hong Kong, Hong Kong, in 2018, and the M.Sc. degree in applied mathematics from Brown University, Providence, RI, USA, in 2019, where he is currently pursuing the Ph.D. degree.

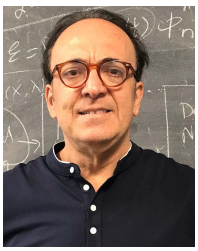
His research interests include scientific machine learning, physics-informed neural networks, and time series modeling.



Ioannis G. Kevrekidis received the bachelor's degree in chemical engineering from the National Technical University of Athens, Athens, Greece, in 1982, and the M.A. degree in mathematics and the Ph.D. degree in chemical engineering from the University of Minnesota, Minneapolis, MN, USA, in 1986.

He was a Director's Post-Doctoral Fellow with the University of New Mexico at Los Alamos, Los Alamos, NM, USA. He was with Princeton University, Princeton, NJ, USA, for 31 years. He moved to the Medical School, Johns Hopkins University, Baltimore, MD, USA, in 2017, where he is currently a Bloomberg Distinguished Professor of chemical and biological engineering, applied mathematics, and statistics. He holds the Emeritus Status at Princeton. He has pioneered the equation-free/variable-free multiscale modeling approach to complex systems modeling. His research interests include computational modeling of complex dynamical systems, with emphasis on data-driven, machine-learning-assisted, and multiscale modeling algorithms over the last 30 years.

Dr. Kevrekidis is a member of the National Academy of Engineering, the American Academy of Arts and Sciences, and the Academy of Athens. He has been a Packard and a Guggenheim Fellow. His work received awards from the American Institute of Chemical Engineers (AIChE), including Colburn, Wilhelm, CAST, and Society for Industrial and Applied Mathematics (SIAM), including the Crawford prize, W. T., and the Idalia Reid Medal.



George Em Karniadakis received the Ph.D. degree in mechanical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1987.

He is currently the Charles Pitts Robinson and John Palmer Barstow Professor of applied mathematics with Brown University, Providence, RI, USA. He has authored four books on spectral elements, microflows and nanoflows, parallel computing, and stochastic partial differential equations. His research interests include various topics related to scientific machine learning.