General Hamiltonian Neural Networks for Dynamic Modeling: Handling Sophisticated Constraints Automatically and Achieving Coordinates Free

Yanfang Liu, Xu Wang, Yituo Song, Bo Wang, and Desong Du, *Graduate Student Member, IEEE*

***Abstract*— Embedding the Hamiltonian formalisms into neural networks (NNs) enhances the reliability and precision of data-driven models, in which substantial research has been conducted. However, these approaches require the system to be represented in canonical coordinates, i.e., observed states should be generalized position-momentum pairs, which are typically unknown. This poses limitations when the method is applied to real-world data. Existing methods tackle this challenge through coordinate transformation or designing complex NNs to learn the symplectic phase flow of the state evolution. However, these approaches lack generality and are often difficult to train. This article proposes a versatile framework called general Hamiltonian NN (GHNN), which achieves coordinates free and handles sophisticated constraints automatically with concise form. GHNN employs two NNs, namely, an HNet to predict the Hamiltonian quantity and a JNet to predict the interconnection matrix. The gradients of the Hamiltonian quantity with respect to the input coordinates are calculated using automatic differentiation and are then multiplied by the interconnection matrix to obtain state differentials. Subsequently, ordinary differential equations (ODEs) are solved by numerical integration to provide state predictions. The accuracy and versatility of the GHNN are demonstrated through several challenging tasks, including the nonlinear simple and double pendulum, coupled pendulum, and real 3-D crane dynamic system.**

***Index Terms*— Data-driven dynamic model, Hamiltonian mechanics, Hamiltonian-inspired framework, physical-informed neural networks (NNs).**

I. Introduction

A. Background and Related Works

ACCURATE dynamic models of mechanical systems are often crucial for predicting system behavior and ensuring the stability of model-based control [1], [2]. Modeling the dynamics of complex systems directly from data has shown significant improvements in model accuracy, offering the potential to endow the system with advanced capabilities [3]. Fully data-driven dynamic models, typically as neural networks (NNs), promise to simplify the process of modeling and analysis. However, the universal approximation theorem, upon which NN models rely, proves that only a sufficiently large network can ensure a small approximation error [4]. They often struggle to generalize to unseen regions of the state space [5], [6]. To enhance accuracy and improve generalization, a promising approach involves integrating data-driven models with the appropriate prior knowledge that aligns with the observed state manifold [7]. Pure NN models often struggle to learn the underlying fundamental principles that govern the dynamic evaluation of physical systems. By embedding the relevant knowledge into NNs, the information content of the data can be substantially amplified, enabling data-driven models to more efficiently converge toward the correct solution [8], [9], [10], [11]. This approach is appealing because it enhances the model's generalization and physical plausibility by integrating knowledge into the regression model [12].

In recent years, the exploration of utilizing NN to learn the fundamental equations of analytical mechanics, particularly focusing on the domains of Lagrangian and Hamiltonian mechanics, represents a significant advancement in physical-informed data-driven methods. Lagrangian mechanics require relatively straightforward data for learning since it defines the equations of motion using state variables and their time derivatives [8], [13]. In contrast, Hamiltonian mechanics require the system to be represented under canonical coordinates to describe more general equations, i.e., the state must be generalized position-momentum pairs. The fundamental Hamiltonian equation is defined as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{q}\\ \boldsymbol{p}\end{bmatrix}=\begin{bmatrix}\mathbf{0} & \boldsymbol{I}\\ -\boldsymbol{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\partial\mathcal{H}/\partial\boldsymbol{q}\\ \partial\mathcal{H}/\partial\boldsymbol{p}\end{bmatrix} \tag{1}$$

where $\boldsymbol{q}$ is the generalized position, and $\boldsymbol{p}$ corresponds to the generalized momentum. Hamiltonian NNs (HNNs) [14] is the most fundamental learning model specific to Hamiltonian systems, which uses a standard NN $\tilde{\mathcal{H}}$ to approximate the Hamiltonian $\mathcal{H}$. HNN operates under the assumption that the data is structured as $\{\boldsymbol{q}, \boldsymbol{p}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{p}}\}$, which must be canonical [15]. In practice, requiring a dynamic system to be represented in canonical coordinates could be too restrictive, especially since obtaining momenta data can be challenging.

Manuscript received 3 October 2023; revised 13 March 2024; accepted 31 May 2024. Date of publication 4 September 2024; date of current version 5 May 2025. This work was supported in part by the National Science Foundation of China under Grant 52272390, in part by the Natural Science Foundation of Heilongjiang Province of China under Grant YQ2022A009, and in part by the State Key Laboratory of Robotics and System (HIT) under Grant SKLRS-2022-ZM-13. *(Corresponding author: Yanfang Liu.)*

Yanfang Liu, Yituo Song, and Desong Du are with the Department of Aerospace Engineering, Harbin Institute of Technology, Harbin 150001, China (e-mail: yanfangliu@hit.edu.cn; 1079851849@qq.com; dudesong@hit.edu.cn).

Xu Wang is with the Department of Aerospace Engineering and the Sate Key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin 150001, China (e-mail: W17749230425@163.com).

Bo Wang is with the National Key Laboratory of Human Factors Engineering, Astronaut Research and Training Center of China, Beijing 100094, China (e-mail: wowbob@139.com).

Digital Object Identifier 10.1109/TNNLS.2024.3409567

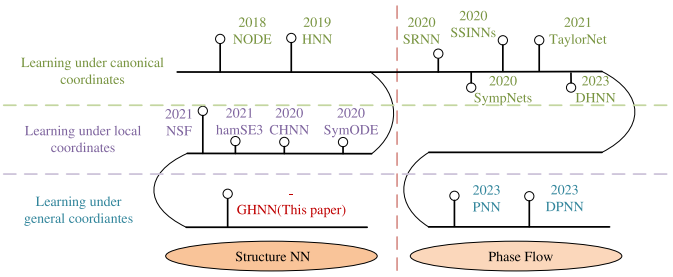


Fig. 1. Comparison of different Hamiltonian-inspired models.

TABLE I
OVERVIEW OF MAIN STREAM STRUCTURE NN-BASED HAMILTONIAN MODELS. GHNN ACHIEVES MANY DESIRABLE PROPERTIES

	HNN	CHNN	NSF	GHNN
Learn from canonical coords	√	√	√	√
Don't require high-order derivative		√		√
Don't require momenta			√	√
On general symplectic forms			√	√
Learn constraints		√		√
Learn from arbitrary coords				√
Validate on real system				√

To overcome this limitation, an alternative approach is to learn the phase flow of the Hamiltonian system directly. This involves encoding physical priors, such as symplecticity, into the flowthrough specifically designed NN architectures. Inspired by this, a series of methods intrinsically embedding a symplectic integrator into the recurrent NN is proposed, such as SRNN [16], [17], [18], [19], [20]. Jin et al. [21] introduced symplectic networks (SympNets) with theoretical guarantees that can approximate arbitrary symplectic maps to capture dynamics for a more general system. Tong et al. [22] proposed a novel NN architecture named symplectic Taylor NNs (TaylorNets), which integrates the Taylor series expansion with a symmetric structure, inherently accommodating the numerical fitting process of the spatial derivatives of the Hamiltonian. Stemming from the discretization of HNNs, Hamiltonian deep NNs (HDNNs) [23] integrate symplectic discretization methods with NN architecture, ensuring the model with nonvanishing gradients. The above methods still suffer from canonical state limitations. Recently, Poisson NNs (PNNs) [24] provide a solution for learning nonsymplectic mechanics. This approach learns to coordinate transformation to canonical coordinate using invertible NNs and makes a prediction in this learned latent space with SympNets [21]. Šípka et al. [25] proposed a robust method of direct PNNs (DPNNs), which can learn dynamics in noncanonical systems without the dimension of the symplectic subsystem. The designed NNs also satisfy Jacobi identity, which is inherent in the data flow. These two works represent significant advancements in achieving the learning of symplectic maps under general coordinates. However, the particular NN architectures are required for the aforementioned phase flow, which is difficult to design.

Another strategy aims to learn the structure of the Hamiltonian systems using the normal deep NNs, called the structure NNs method. This article prefers and focuses on this insight, establishing the model conveniently. In this way, SymODE learns mass matrix $\boldsymbol{M}(\boldsymbol{q})$ by NN to facilitate the conversion between $\dot{\boldsymbol{q}}$ and $\boldsymbol{p}$ in $\boldsymbol{p} = \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$ [26], which relaxes the limitation the momentum. SymODE also incorporates a specific interconnection matrix tailored to accommodate embedded joint angles coordinate, which has the state $[\cos \boldsymbol{q}, \sin \boldsymbol{q}]^{\mathrm{T}}$. Furthermore, Duong and Atanasov [27] proposed a learned Hamiltonian formulation applied to the $SE(3)$ manifold. It is utilized to approximate the dynamics of a rigid body, where attitude angles are represented using rotation matrix [28]. To effectively handle the dynamics of the constrained systems using data-driven models, constrained HNNs (CHNNs) represent the system using Cartesian coordinates and enforce the holonomic constraints [29] explicitly by Lagrange multipliers [30]. However, this method relies on the prior knowledge of the constraints, and the complex constraint matrix needs to be hand-designed. Neural symplectic form (NSF) [31] employs an NN to learn the differential one-form, which is then subjected to an exterior derivative to give the symplectic two-form. This extension allows it to handle the dynamics of a generalized symplectic two-form Hamiltonian system.

However, structure NN models are inherently constrained by the choice of coordinates. The aforementioned approaches manually derive specific cases, but are not applicable to general coordinates [32]. Therefore, this article focuses on the above problem, aiming to extend structure NN models to handle arbitrary observations.

B. Limitations of the Previous Works

Several powerful methods are introduced for learning Hamiltonian-inspired data-driven models, as summarized in Fig. 1. Based on particular frameworks or explicitly known constraints, structure NN methods have demonstrated outstanding results in specific coordinates, as illustrated in Table I, but certain limitations persist. This article concentrates on structure NNs methods, which limitations are as follows.

1) *Coordinate Limitation:* Existing models rely on functional relationships derived from Hamiltonian mechanics, inherently tied to specific coordinates, and struggle to generalize to other coordinates, e.g., HNNs primarily operate in the canonical coordinate.
2) *Requirement for Extra Information:* Existing models often require high-order data derivatives or even knowledge of the true Hamiltonian. This dependency limits their effectiveness in learning the dynamics of practical systems.
3) *High Computation Complexity:* The computational graph of physical-informed NN frameworks is complex, making its gradient descent challenging. This may lead to difficulties in model training and bring prediction divergence.

Notation: Under the same NN structure and training hyperparameters, incorporating more physical laws leads to a more general, plausible, and stable data-driven model. However, more physical laws usually impose more limitations on the model's applicability. In addition, the intricate model structure

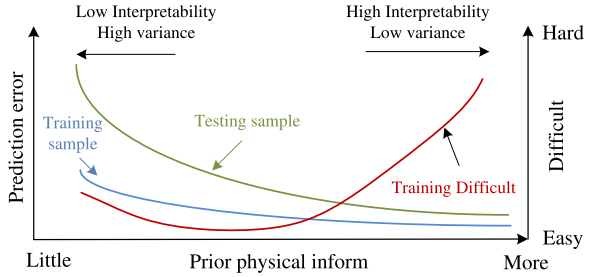


Fig. 2. Prediction error and training difficulty with respect to informed physical laws.

of complex physical laws leads to difficulties in model training and brings time-costing for prediction, as shown in Fig. 2. Above all, it's critical to balance the tradeoff between physical laws and the feasibility of training. Consequently, there's a clear requirement for an appropriate framework and clever combination with NNs to create a versatile learning model.

C. Contributions and Novelties

Theoretically, expressing dynamic systems using Hamiltonian mechanics under noncanonical coordinates is quite challenging. For instance, learning dynamics of systems described under position-velocity pairs need to additional mass matrix. One may notice that system dynamics can be represented by the Hamiltonian equations under special reasonable coordinates by employing a particular skew-symmetric matrix $\boldsymbol{J}$. Following this motivation, this article proposes a general HNN (GHNN) that focuses on learning dynamics without being restricted by coordinate limitations by directly learning the matrix $\boldsymbol{J}$ from data. The primary framework and capacity of the GHNN are displayed in Fig. 3. Unlike previous works, GHNN only relies on Hamiltonian nature without any additional prior information about the dynamics.

In particular, this article makes the following contributions.

1) Introducing a novel Hamiltonian framework for dynamic systems in general coordinates.
2) Developing NNs with the proposed general Hamiltonian framework to learn dynamics from data efficiently, achieve coordinate free and handle complex constraints automatically.
3) Designing a novel training approach to ensure robust convergence, incorporating an efficient loss function that covers both accuracy and energy conservation.
4) Phase figures for simulations and time domain plots for experiments are reported and compared in various cases.

Notation: This article uses $\hat{\cdot}$ to denote the prediction state. The canonical state is represented as $z = [\boldsymbol{q}^{\mathrm{T}}, \boldsymbol{p}^{\mathrm{T}}]^{\mathrm{T}}$, where $\boldsymbol{q} \in \mathbf{R}^n$ and $\boldsymbol{p} \in \mathbf{R}^n$ denote the general position and momenta, respectively. Conversely, the configuration vector in a local coordinate is denoted as $\boldsymbol{y} = [\boldsymbol{u}^{\mathrm{T}}, \boldsymbol{v}^{\mathrm{T}}]^{\mathrm{T}}$, with $\boldsymbol{u} \in \mathbf{R}^{m_q}$ and $\boldsymbol{v} \in \mathbf{R}^{m_p}$ representing the local position and momenta, respectively. The inner-connection matrix in the canonical coordinate, which is in standard form, is denoted by $\boldsymbol{J}$, while its general form in a local coordinate is represented by $\bar{\boldsymbol{J}}$.

II. Preliminaries and Hamiltonian Mechanics

A. Hamiltonian Mechanics

Hamiltonian dynamics reformulates Newtonian dynamics by describing a physical system using generalized position and momenta. It offers profound insights into the mechanics laws in a symplectic form, with definitions as follows.

Definition 1: The canonical Hamiltonian system takes the form

$$\dot{z} = \boldsymbol{J}\nabla\mathcal{H}(z) \tag{2}$$

where $\boldsymbol{J}$ is the inner-connection matrix, and is skew-symmetric [15], [33]

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \in \mathbf{R}^{2n\times 2n} \tag{3}$$

where $\boldsymbol{I}$ is the identity matrix with proper dimensions.

B. Hamiltonian Mechanics With Only Velocity Observed

The generalized momenta is often unavailable in the practical environment. Instead, the generalized velocity $\boldsymbol{\xi}$ is easy to obtain. The generalized momenta $\boldsymbol{p}$ can be obtained as

$$\boldsymbol{p} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{\xi} \tag{4}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbf{R}^{n\times n}$ is a symmetric positive mass matrix. The Hamiltonian usually takes the form of

$$\mathcal{H} = \frac{1}{2}\boldsymbol{p}\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{p} + \boldsymbol{V}(\boldsymbol{q}). \tag{5}$$

Thus, the system is transformed in canonical coordinates, with dynamic motion satisfies (2). The time derivative of the generalized velocity has

$$\dot{\boldsymbol{\xi}} = \left(\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{M}^{-1}(\boldsymbol{q})\right)\boldsymbol{p} + \boldsymbol{M}^{-1}(\boldsymbol{q})\dot{\boldsymbol{p}}. \tag{6}$$

Several works handle the system with implicit momenta by learning the mass matrix. However, this requires additional NNs to approximate different parts of the (5), making the training process difficult [26], [27].

C. Hamiltonian Mechanics Under Local Coordinates

Sections II-A and II-B exhibit that using generalized position with corresponding momenta or velocity, the systems are consistently able to be described by Hamiltonian formalism. However, other local coordinates are utilized to overcome the limitation of canonical coordinates in practice. For example, continuous revolute joints represented by the canonical angle are problematic due to angle wrapping at $\pm\pi$. This issue is often addressed using sine/cosine feature transformations. In addition, the attitude of aircraft is usually expressed using quaternions or rotation matrix, instead of Euler angles to avoid the singularity. These coordinates help mitigate the nonlinearity of the dynamic equations and facilitate downstream controller design. Existing works demonstrate that some of the local coordinates can also be described by (2), with a particular form of $\boldsymbol{J}$ [28].

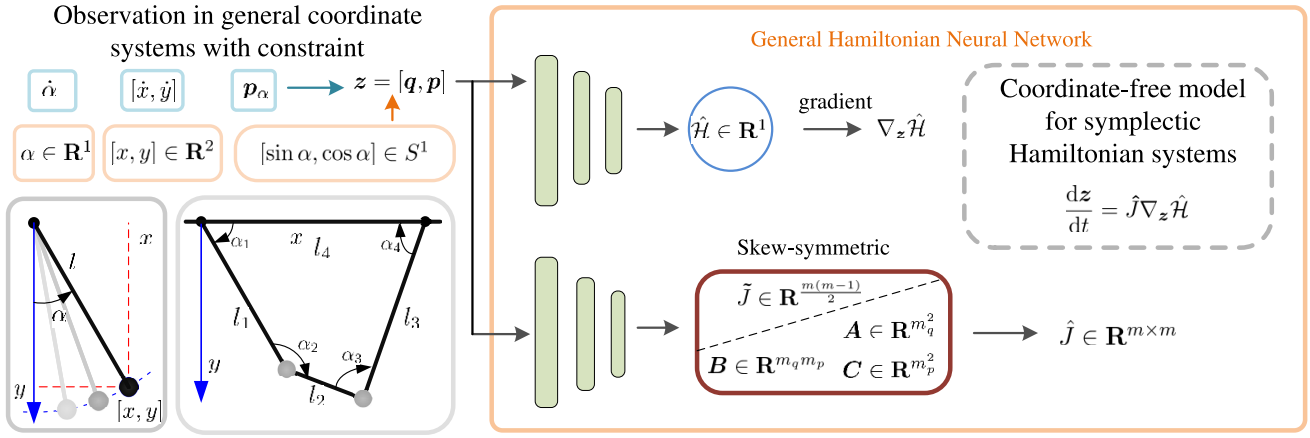


Fig. 3. Framework of the GHNN.

For example, if the pendulum depicted in Fig. 3 has an embedding form state $\boldsymbol{y} = [\cos\alpha, \sin\alpha, p_\alpha]^{\mathrm{T}}$, the interconnection matrix takes the form as [26]

$$\boldsymbol{J}_e = \begin{bmatrix} 0 & 0 & -\sin\alpha \\ 0 & 0 & \cos\alpha \\ \sin\alpha & -\cos\alpha & 0 \end{bmatrix}. \tag{7}$$

In the Cartesian coordinate with the local state $\boldsymbol{y} = [x, y, p_x, p_y]^{\mathrm{T}}$, the matrix is [30]

$$\boldsymbol{J}_c = \begin{bmatrix} 0 & 0 & 1 - \frac{x^2}{l^2} & -\frac{xy}{l^2} \\ 0 & 0 & -\frac{xy}{l^2} & 1 - \frac{y^2}{l^2} \\ \frac{x^2}{l^2} - 1 & \frac{xy}{l^2} & 0 & \frac{p_x y - p_y x}{l^2} \\ \frac{xy}{l^2} & \frac{y^2}{l^2} - 1 & \frac{p_y x - p_x y}{l^2} & 0 \end{bmatrix}. \tag{8}$$

D. Hamiltonian Mechanics With Constraints

For the coupled pendulum depicted in Fig. 3, the standard Hamiltonian equation is insufficient to represent its dynamics, since the closed-loop constraints must be considered. Thus, in Cartesian coordinates, with the system be described as $\boldsymbol{y} = [\boldsymbol{u}_1^{\mathrm{T}}, \ldots, \boldsymbol{u}_n^{\mathrm{T}}, \boldsymbol{v}_1^{\mathrm{T}}, \ldots, \boldsymbol{v}_n^{\mathrm{T}}]$, where $\boldsymbol{u}_i = [x, y]^{\mathrm{T}}$, $\boldsymbol{v}_i = [p_x, p_y]^{\mathrm{T}}$, the connections between point masses must adhere to rigidity constraints

$$\|\boldsymbol{u}_1 - \boldsymbol{u}_2\| - l_2^2 = 0 \tag{9}$$

where $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are the position vectors of two adjacent masses in the world inertia coordinate, and l_2 denotes the corresponding link length. As a rigidity constraint, the differential of the equation must also follow the constraints, which has:

$$2(\boldsymbol{q}_1 - \boldsymbol{q}_2)^{\mathrm{T}}(\dot{\boldsymbol{q}}_1 - \dot{\boldsymbol{q}}_2) = 0. \tag{10}$$

Remark 1: The Hamiltonian equation with holonomic constraints is commonly used to describe closed-loop systems. The constraints are described as

$$\begin{aligned} &\phi(\boldsymbol{q}) = \boldsymbol{0} \\ &\dot{\boldsymbol{\phi}}(\boldsymbol{q}) = \nabla_q \boldsymbol{\phi}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{q})\nabla_{\boldsymbol{p}}\mathcal{H}(z) = \boldsymbol{0} \end{aligned} \tag{11}$$

where $\boldsymbol{N}(\boldsymbol{q}) = \nabla_q \boldsymbol{\phi}(\boldsymbol{q}) \in \mathbf{R}^{r \times m}$ is Jacobian matrix, and r is the constraint dimension.

Then, (2) can be rewritten as

$$\dot{\boldsymbol{z}} = [\boldsymbol{J} - \boldsymbol{J}\boldsymbol{P}^{\mathrm{T}}[\boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{\mathrm{T}}]^{-1}\boldsymbol{P}\boldsymbol{J}]\nabla_z \mathcal{H}(z) \tag{12}$$

where $\boldsymbol{P} = \nabla_z([\boldsymbol{\phi}(\boldsymbol{q}), \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{q})\nabla_{\boldsymbol{p}}\mathcal{H}(z)])$ [30].

CHNN follows this framework by setting the constraint as (11) explicitly and using (12) to learn the system dynamics in Cartesian coordinate. But this requires the constraints to be known and the training process is challenging. Details will be further discussed in Section IV-D.

E. Coordinate Changes and the Darboux-Lie Theorem

Theorem 1: (Darboux 1882, Lie 1888). Suppose that the system represent by the state $\boldsymbol{y} = [\boldsymbol{u}^{\mathrm{T}}, \boldsymbol{v}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbf{R}^m$ and is of constant rank $r = m - 2n$. Then, there exist functions $\boldsymbol{P}(\boldsymbol{y}) : \mathbf{R}^m \to \mathbf{R}^n$, $\boldsymbol{Q}(\boldsymbol{y}) : \mathbf{R}^m \to \mathbf{R}^n$ and $\boldsymbol{C}(\boldsymbol{y}) : \mathbf{R}^m \to \mathbf{R}^r$ satisfying

$$\begin{aligned} &\{P_i, P_j\} = 0\{P_i, Q_j\} = -\delta_{ij}\{P_i, C_l\} = 0 \\ &\{Q_i, P_j\} = \delta_{ij}\{Q_i, Q_j\} = 0\{Q_i, C_l\} = 0 \\ &\{C_k, P_j\} = 0\{C_k, Q_j\} = 0\{C_k, C_l\} = 0 \end{aligned} \tag{13}$$

where δ_{ij} equals to 1 if $i = j$ else 0. $\{\cdot, \cdot\}$ is the Possion bracket which has

$$\{\boldsymbol{F}, \boldsymbol{G}\} = \sum_{i=1}^{d}\left(\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}_i}\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{v}_i} - \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}_i}\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{v}_i}\right) = \nabla \boldsymbol{F}(\boldsymbol{y})^{\mathrm{T}}\boldsymbol{J}\nabla \boldsymbol{G}(\boldsymbol{y}). \tag{14}$$

The gradients of P_i, Q_j, and C_k are linearly independent so that $\boldsymbol{y} \to (P_i(\boldsymbol{y}), Q_j(\boldsymbol{y}), C_k(\boldsymbol{y}))$. constitutes a transformation from a local coordinate to the canonical.

The dynamics of practical systems need to be represented in a reasonable coordinate. Thus, a hypothesis gives that the general position of canonical coordinates can be viewed as a mapping of the position from local coordinates. Leveraging this insight, the following corollary is derived.

Corollary 1 (Transformation to Canonical Form): Supposing the system in a local coordinate has state $\boldsymbol{y} = [\boldsymbol{u}^{\mathrm{T}}, \boldsymbol{v}^{\mathrm{T}}]^{\mathrm{T}}$, as $\boldsymbol{u} \in \mathbf{R}^{m_q}, \boldsymbol{v} \in \mathbf{R}^{m_p}$ is a general

position and momenta in the local coordinate, respectively. Here denoting the transformation of Theorem 1 by $z = [q^{\mathrm{T}}, p^{\mathrm{T}}]^{\mathrm{T}} = [P(u), Q(u, v)]$. Thus, the system represented in local coordinate with state y can be transformed to canonical coordinate with

$$\dot{z} = J\nabla\bar{\mathcal{H}}(z) \tag{15}$$

where $\bar{\mathcal{H}}(z)$ is the Hamiltonian in the transformed canonical coordinate.

F. Learning Dynamic From Data

Assuming that the dynamics of a physical system is governed by ordinary differential equations (ODEs) $\dot{y} = f(y) : \mathbf{R}^m \mapsto \mathbf{R}^m$, where $y \in \mathbf{R}^m$ is the system state, and f is the dynamics of the system.

This article focuses on learning the dynamics f directly from observed time series data $Y = (y_{t_1}, y_{t_2}, \ldots, y_{t_N})$. Subsequently, with the learned dynamics $\hat{f}$, it aims to predict long term behavior $\hat{Y} = (\hat{y}_{t_1'}, \hat{y}_{t_2'}, \ldots, \hat{y}_{t_N'})$ from any given initial state $\hat{y}_{t_0'}$.

Following the neural ODEs [34] approach, the system dynamics is approximated by a NN $\hat{f}_\theta$ with parameters θ. Consequently, the predicted states are calculated by ODE solvers, which has

$$\begin{aligned}\hat{z}_{t_{i+1}} &= \int_{t_i}^{t_{i+1}} \hat{f}_\theta(\hat{z}_{t_i})\mathrm{d}t + \hat{z}_{t_i}, \quad \text{s.t. } \hat{z}_{t_0} = z_{t_0} \\ &= \mathrm{ODESolver}(\hat{z}_{t_i}, \hat{f}_\theta, t).\end{aligned} \tag{16}$$

The NN is training by computing the gradients of the loss function using ordinary backpropagation or the adjoint method [34].

The critical factor is how to design the structure of $\hat{f}_\theta$ to approximate the dynamics f with appropriate inductive biases. Section III will give a detailed illustration of the proposed method.

III. Methods

In a practical dynamic system, the observed states may be represented in arbitrary reasonable coordinates due to the nature of measurement equipment. Therefore, an ideal Hamiltonian-inspired data-driven model should be coordinates-free and straightforward to construct and train.

A. General Hamiltonian Mechanics

Theorem 1 illustrates there always exist reversible transformations to map local coordinates into the canonical coordinate. In addition, local coordinates also include the coordinate represented by position-velocity pairs (4).

Proposition 1: Considering a physical system represented in a reasonable local coordinate $y = [u^{\mathrm{T}}, v^{\mathrm{T}}]^{\mathrm{T}}$, the dynamics can be reformulated (2) as

$$\begin{bmatrix}\dot{u} \\ \dot{v}\end{bmatrix} = \bar{J}(u, v)\begin{bmatrix}\nabla_u\bar{\mathcal{H}}(u, v) \\ \nabla_v\bar{\mathcal{H}}(u, v)\end{bmatrix} \tag{17}$$

and the matrix $\bar{J}$ maintains skew-symmetric.

Proof: The state of a dynamic system described in a local coordinate can be seen as a mapping from the canonical coordinate. Thus, here denoting the transformation of Corollary 1 as

$$u = Q(q) \in \mathbf{R}^{m_q}, \quad v = P(q, p) \in \mathbf{R}^{m_p} \tag{18}$$

where the function $Q : \mathbf{R}^n \to \mathbf{R}^{m_q}$ represents the position transformation, and $P : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^{m_p}$ is the momentum transformation. Then, one has

$$\bar{\mathcal{H}}(u, v) = \mathcal{H}(q, p). \tag{19}$$

The state deferential has

$$\begin{aligned}\dot{u} &= A\dot{q} \\ \dot{v} &= B\dot{p} + C\dot{q}\end{aligned} \tag{20}$$

where $A = \nabla_q Q(q)$, $B = \nabla_p P(q, p)$, and $C = \nabla_q P(q, p)$ are the Jacobian matrices, respectively.

Substituting (2) and (19) into (20) gives

$$\begin{aligned}\dot{u} = A\dot{q} &= A\nabla_p\mathcal{H}(q, p) \\ &= A\nabla_p^{\mathrm{T}} v\nabla_v\bar{\mathcal{H}}(u, v) \\ &= A\nabla_p^{\mathrm{T}} G(q, p)\nabla_v\bar{\mathcal{H}}(u, v) \\ &= AB^{\mathrm{T}}\nabla_v\bar{\mathcal{H}}(u, v)\end{aligned} \tag{21}$$

and for the derivation of the momenta, it has

$$\begin{aligned}\dot{v} &= B\dot{p} + C\dot{q} \\ &= -B\nabla_q\mathcal{H}(q, p) + C\nabla_p\mathcal{H}(q, p) \\ &= -B\left[\nabla_q^{\mathrm{T}} u\nabla_u\bar{\mathcal{H}}(u, v) + \nabla_q^{\mathrm{T}} v\nabla_v\bar{\mathcal{H}}(u, v)\right] \\ &\quad + C\nabla_p^{\mathrm{T}} v\nabla_v\bar{\mathcal{H}}(u, v) \\ &= -BA^{\mathrm{T}}\nabla_u\bar{\mathcal{H}}(u, v) + (CB^{\mathrm{T}} - BC^{\mathrm{T}})\nabla_v\bar{\mathcal{H}}(u, v).\end{aligned} \tag{22}$$

Then, the dynamics under the local coordinate has

$$[] = \begin{bmatrix}0 & AB^{\mathrm{T}} \\ -BA^{\mathrm{T}} & CB^{\mathrm{T}} - BC^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\nabla_u\bar{\mathcal{H}}(u, v) \\ \nabla_v\bar{\mathcal{H}}(u, v)\end{bmatrix}. \tag{23}$$

Thus,

$$\bar{J}(q, p) = \begin{bmatrix}0 & AB^{\mathrm{T}} \\ -BA^{\mathrm{T}} & CB^{\mathrm{T}} - BC^{\mathrm{T}}\end{bmatrix}. \tag{24}$$

And using inverse transformation, there has

$$q = \bar{Q}(u) \in \mathbf{R}^n, \quad p = \bar{P}(u, v) \in \mathbf{R}^n. \tag{25}$$

Substituting the transform (25) into (23) can represent J in the local coordinate

$$\bar{J}(q, p) = \bar{J}(\bar{Q}(u), \quad \bar{P}(u, v)) = \bar{J}(u, v). \tag{26}$$

Obviously, the matrix $\bar{J}(u, v)$ is skew-symmetric. ■

Remark 2: Proposition 1 demonstrates that the system dynamics also has a Hamiltonian form in a local coordinate, with the interconnection matrix $J(u, v)$ has a certain form and still holds skew-symmetric. In addition, if constraints are present, they can also be implicitly satisfied by the general form of matrix $J(u, v)$ in (23).

Remark 3: Actually, the skew-symmetric property of the $\bar{\boldsymbol{J}}$ can also be verified straightforwardly by considering the energy conservation along a trajectory. Since

$$\dot{\mathcal{H}}(\boldsymbol{y}) = (\nabla_{\boldsymbol{y}}\mathcal{H}(\boldsymbol{y}))^{\mathrm{T}}\dot{\boldsymbol{y}} = (\nabla_{\boldsymbol{y}}\mathcal{H}(\boldsymbol{y}))^{\mathrm{T}}\boldsymbol{J}\nabla_{\boldsymbol{y}}\mathcal{H}(\boldsymbol{z}) = 0 \quad (27)$$

holds if and only if $\bar{\boldsymbol{J}}$ is skew-symmetric.

Finding the transformation from an arbitrary local coordinate to a canonical coordinate remains challenging, and existing methods also struggle to automatically determine such a representation. In another field, NNs offer an alternative way to learning the general $\bar{\boldsymbol{J}}$ directly from data.

B. General HNNs

As illustrated in Section III-B, system dynamics in any reasonable local coordinates have the general Hamiltonian form (17). Building upon this formulation, this article proposes the GHNN framework, which implicitly handles coordinate transformations by learning complex skew-symmetric matrix $\boldsymbol{J}$ through an individual NN. The computational graph is illustrated in Fig. 4.

1) Primary Architecture: The proposed framework consists of two NNs: the JNet and the HNet. Both NNs take the states $\boldsymbol{y}$ as input, with the JNet producing the matrix $\boldsymbol{J}$, and the HNet generating the Hamiltonian $\mathcal{H}$. In-graph gradients of $\mathcal{H}$ with respect to the input coordinates $\boldsymbol{y}$ are calculated by automatic differentiation. The obtained $\nabla_{\boldsymbol{y}}\mathcal{H}$ is further multiplied by $\boldsymbol{J}$ to determine the change rate of the states $\dot{\boldsymbol{y}}$. This rate is subsequently integrated using an ODE solver to give the prediction of the coming state. Both NNs have input dimension m. The output of the HNet is a scalar. For the JNet, noting the fact that $\boldsymbol{J}$ is a skew-symmetric matrix, so only the upper or lower triangular matrix (excluding the diagnose elements, which are always 0) needs to be learned. Therefore, the output dimensions of JNet are $((m^2 - m)/2)$.

2) Alternative Architecture: This approach introduces a decomposite GHNN (DGHNN) by exploiting a further complex component of the $\boldsymbol{J}$. In DGHNN, the matrix $\bar{\boldsymbol{J}}$ is decomposed into matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ as (24). Consequently, JNet is partitioned into three subnetworks named ANet, BNet, and CNet, each tasked with approximating the corresponding matrix.

The input of ANet is the position state $\boldsymbol{u}$ with dimension m_q, and it outputs the element of matrix $\boldsymbol{A}$ with dimension $m_q n$, denoted as $\boldsymbol{f}_A(\boldsymbol{u}) : \mathbf{R}^{m_q} \to \mathbf{R}^{m_q n}$. Similarly, BNet and CNet have $\boldsymbol{f}_B(\boldsymbol{y}) : \mathbf{R}^{m} \to \mathbf{R}^{m_p n}$, $\boldsymbol{f}_C(\boldsymbol{y}) : \mathbf{R}^{m} \to \mathbf{R}^{m_p n}$, respectively. Then, the $\bar{\boldsymbol{J}}$ can be obtained using (24).

By embedding the Hamiltonian equations into NNs, the prior knowledge of Hamiltonian dynamics is incorporated into data-driven models in a systematic way. Consequently, GHNN has

$$\hat{\dot{\boldsymbol{y}}} = \hat{\boldsymbol{J}}(\boldsymbol{y}, \boldsymbol{\theta}_1)\nabla_{\boldsymbol{z}}\hat{\mathcal{H}}(\boldsymbol{y}, \boldsymbol{\theta}_2) = \hat{\boldsymbol{f}}(\boldsymbol{y}, \boldsymbol{\theta}) \quad (28)$$

where $\hat{\boldsymbol{J}}(\boldsymbol{y}, \boldsymbol{\theta}_1)$ is the output of the JNet with parameters $\boldsymbol{\theta}_1$ used to approximate the skew-symmetric $\boldsymbol{J}$, and $\hat{\mathcal{H}}(\boldsymbol{y}, \boldsymbol{\theta}_2)$ corresponds to the output of the HNet with parameters $\boldsymbol{\theta}_2$ to represent Hamiltonian $\mathcal{H}$. Consequently, the general Hamiltonian differential equation for arbitrary coordinate systems based on NNs with parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2]$ is derived.

In such a coordinate-free framework, the most suitable coordinate can be chosen for the task. This flexibility also allows GHNN to adapt to the local coordinates limited by practical sensors. Importantly, GHNN learns the constraints from data implicitly, eliminating the need for explicit knowledge of the constraints. The only requirement is that the dynamics of the system are fully governed by the chosen states, indicating that the system must be entirely observable.

For the forward prediction, as described in (16), starting from an initial state $\boldsymbol{y}_{t_0}$, the predictions of subsequent states $\{\hat{\boldsymbol{y}}_{t_1}, \hat{\boldsymbol{y}}_{t_2}, \ldots, \hat{\boldsymbol{y}}_{t_N}\}$ can be obtained through an iterative manner. When observations $\{\boldsymbol{y}_{t_1}, \boldsymbol{y}_{t_2}, \ldots, \boldsymbol{y}_{t_N}\}$ are available, parameters of both NNs can be learned by minimizing the distance between observations with predictions. To effectively determine these parameters, this article introduces a specific loss function and training process, details are provided in Section III-C.

C. Loss Function Design

Considering the Hamiltonian equation in (28), the prediction state corresponding to time series $\boldsymbol{t} = [t_0, t_1, t_2, \ldots, t_N]$ can be calculated by (16), where N is the length of the trajectory. Solving the integration by an ODE solver, it has

$$[\hat{\boldsymbol{y}}_{t_0}, \hat{\boldsymbol{y}}_{t_1}, \ldots, \hat{\boldsymbol{y}}_{t_N}] = \mathrm{ODESolve}(\boldsymbol{y}_{t_0}, \hat{\boldsymbol{f}}(\boldsymbol{y}, \boldsymbol{\theta}), \boldsymbol{t}). \quad (29)$$

The parameters $\boldsymbol{\theta}$ of the GHNN $\hat{\boldsymbol{f}}(\boldsymbol{y}, \boldsymbol{\theta})$ can be trained using a dataset $\mathcal{D} = \{\boldsymbol{t}^k_{0:N}, \boldsymbol{y}^k_{t_{0:N}}\}_{k=1:D}$ consisting of state trajectory samples. These samples are collected over a certain time interval from the system, where k is the trajectory index and D is the total number of trajectories. The prediction loss function is given by

$$\mathcal{L}_{\mathrm{pred}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{MNm}\sum_{k=1}^{M}\sum_{i=1}^{N}\|\hat{\boldsymbol{y}}^k_{t_i} - \boldsymbol{y}^k_{t_i}\|_1 \quad (30)$$

where M is the size of the mini-batch.

However, the above loss function is not ideal for learning unknown $\mathcal{H}$ and $\boldsymbol{J}$. The reason is that the Hamiltonian equation is invariant to linear transformation, making GHNN do not have a certain resolution with loss (30).

Remark 4: Supposing the $\mathcal{H}$ and $\boldsymbol{J}$ are the truth Hamiltonian quantity and interconnection matrix, the $\mathcal{H}_\beta = \mathcal{H}/\beta + \gamma$ and $\boldsymbol{J}_\beta = \beta\boldsymbol{J}$ are also reasonable as

$$\dot{\boldsymbol{y}} = \boldsymbol{J}\nabla_{\boldsymbol{y}}\mathcal{H} = \boldsymbol{J}_\beta/\beta\nabla_{\boldsymbol{y}}(\mathcal{H}_\beta\beta - \gamma\beta) = \boldsymbol{J}_\beta\nabla_{\boldsymbol{y}}\mathcal{H}_\beta \quad (31)$$

where β is a constant scale and γ is a constant bias.

Therefore, the above simple end-to-end training framework for GHNN presents two challenges as follows.

1) The training of $\mathcal{H}$ and $\boldsymbol{J}$ is coupled, making it difficult for GHNN to learn these two components from a single end-to-end loss. To illustrate, it's akin to calculating the unknown value a and b when only knows their product $a \cdot b = 1$.
2) The $\mathcal{H}$ and $\boldsymbol{J}$ have a convergence region instead of a single convergence point, which means they have infinity available solution, as discussed in Remark 4. This introduces significant challenges during training,

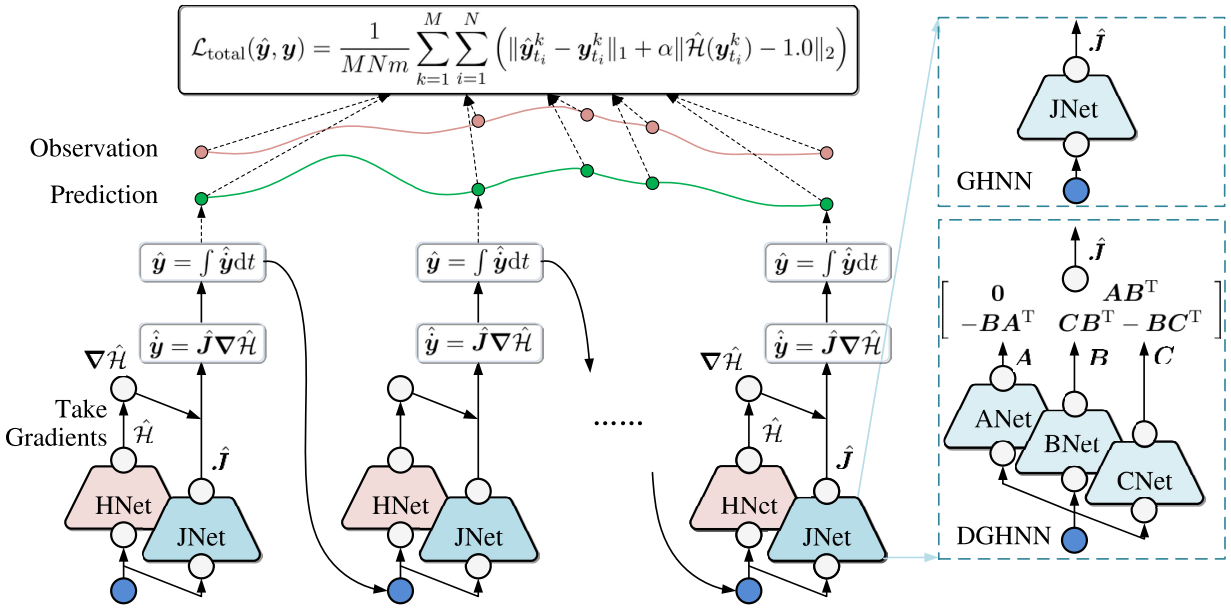


Fig. 4. Computational graph of the GHNN.

as the learned values of H and J may fluctuate within the suboptimal region and are difficult to escape.

However, on the flip side, the absence of a certain solution for $\mathcal{H}$ implies that any value for $\mathcal{H}$ is potentially valid. It's worth noticing that within a single trajectory, the total energy $\mathcal{H}$ remains conserved, which has

$$\mathcal{H}(\boldsymbol{y}_{t_0}^k) = \mathcal{H}(\boldsymbol{y}_{t_1}^k) = \cdots = \mathcal{H}\left(\boldsymbol{y}_{t_N}^k\right). \tag{32}$$

Therefore, a natural approach is to set $\mathcal{H}$ to be constant for the same trajectory.[1]

To achieve energy conservation explicitly and prevent the HNet from being lazy, this article introduces an energy loss function as

$$\mathcal{L}_{\text{energy}}(\hat{\boldsymbol{z}}, \boldsymbol{z}) = \frac{1}{MNm}\sum_{k=1}^{M}\sum_{i=1}^{N}\left\|\hat{\mathcal{H}}(\boldsymbol{y}_{t_i}^k) - c\right\|_2 \tag{33}$$

where c is the constant that needs to be selected, and it can be straightforwardly set to $c = 1$. It can effectively ensure the satisfaction of (32). Furthermore, this loss term also enables HNet to learn energy variations of different trajectories within upbounds.

Proposition 2: For learned $\hat{\mathcal{H}}$ which has a scale and bias, the energy difference of the energy loss term (33) across different trajectories has

$$\left|\|\hat{\mathcal{H}}(\boldsymbol{y}_{t_i}^m) - c\| - \|\hat{\mathcal{H}}(\boldsymbol{y}_{t_j}^n) - c\|_1\right| \le \frac{E}{\beta} \quad \forall m,\ n \in [0, D] \tag{34}$$

where E denotes the truth energy bounded within the system.

Proof: Notice that for autonomous systems, the energy is bounded. Thus, for different trajectories, the following relationship holds:

$$\|\mathcal{H}(\boldsymbol{y}_{t_i}^m) - \mathcal{H}(\boldsymbol{y}_{t_j}^n)\|_1 \le E \quad \forall\, m, n \in [0, D]. \tag{35}$$

Consequently, for any two trajectories, the energy divergence with a constant c is satisfied

$$\begin{aligned}&\left||\hat{\mathcal{H}}\left(\boldsymbol{y}_{t_i}^k\right) - c| - |\hat{\mathcal{H}}\left(\boldsymbol{y}_{t_j}^k\right) - c|\right| \\ &= \left||\mathcal{H}/\beta\left(\boldsymbol{y}_{t_i}^k\right) + \gamma - c| - |\mathcal{H}/\beta\left(\boldsymbol{y}_{t_j}^k\right) + \gamma - c|\right| \\ &\le \frac{1}{\beta}\left|\mathcal{H}\left(\boldsymbol{y}_{t_i}^k\right) - \mathcal{H}\left(\boldsymbol{y}_{t_j}^k\right)\right| \\ &\le \frac{E}{\beta}.\end{aligned} \tag{36}$$

As a result, if β is sufficiently large, the energy loss is bounded with a small range. ■

Then, the total loss function for GHNN is designed as

$$\begin{aligned}&\mathcal{L}_{\text{total}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \\ &= \mathcal{L}_{\text{pred}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \alpha\mathcal{L}_{\text{energy}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \\ &= \frac{1}{MNm}\sum_{k=1}^{M}\sum_{i=1}^{N}\left(\|\hat{\boldsymbol{y}}_{t_i}^k - \boldsymbol{y}_{t_i}^k\|_1 + \alpha\|\hat{\mathcal{H}}(\boldsymbol{y}_{t_i}^k) - 1.0\|_2\right)\end{aligned} \tag{37}$$

where α is the parameter used to balance the tradeoff between the two loss terms. By choosing an appropriate value for α, this loss function ensures GHNN achieves precise trajectory predictions and also informs the HNet to explicitly learn the energy conservation in the system.

The gradient of the loss function is directly backward propagated through the differential ODE solver. Consequently, the parameters $\boldsymbol{\theta}$ are updated using gradient descent to minimize the loss.

D. Random Resampled Method for the Dataset Enhancement

The primary objective in model learning is to ensure robust performance, meaning that the model should deliver consistence results across various runs, even when trained on

[1]To preserve the conservation of $\hat{\mathcal{H}}$, one might consider minimize the variance of the $\mathcal{H}$ within a single trajectory, using a loss function as $\mathcal{L} = \text{var}[\hat{\mathcal{H}}(\boldsymbol{y}_{t_{0:N}}^k)]$. However, this approach is ineffective, as it can be trivially minimized by setting $\hat{\mathcal{H}}(\boldsymbol{y}_{t_{0:N}}^k) = \boldsymbol{0}$.

different datasets. In addition, the learned model should be accurate and robust enough to facilitate long-term predictions. To achieve these goals, it is crucial to utilize as much information as possible information contained in the dataset.

This article proposes a random time interval resampling method to address this issue. Assuming a dataset with the fixed time interval h contains the state in time series $\boldsymbol{t} = \{h, 2h, 3h, \ldots, Nh\}$. The training process involves using a random integer coefficient, denoted as e, to resample the states from the dataset trajectory with length I. Consequently, the resampling training time series becomes $\boldsymbol{t}_r = \{eh, 2eh, 3eh, \ldots, Ieh\}$ with Ie $< N$.

This approach divides the dataset into various time sequences and resamples them with different time steps, resulting in a smoother training process. A further analyze can be found in the Appendix.

Algorithm 1 Training process of GHNN

1: Collect sufficient data to generate dataset.
2: Resample the dataset with random skip interval $c \sim U(1, 10)$ and a forward step I.
3: Construct HNet and JNet. Set training episodes l_1.
4: **for** $i = 1, 2, \ldots, l_1$ **do**
5: Randomly select M amount of training trajectory samples and feed them into JNet and HNet.
6: Output the $\hat{\mathcal{H}}$ and $\hat{\boldsymbol{J}}$ by HNet and JNet, respectively.
7: Calculate the state differential by (28).
8: Prediction the states at the time points $\boldsymbol{t}_r$ by (16).
9: Calculate loss by (37).
10: Train the network by minimizing the loss.
end for

E. Computational Complexity

The computational complexity of GHNN and DGHNN primarily arises from the matrix multiplication and computation of gradients of the Hamiltonian quantity. Suppose each NN has L hidden layers, with each layer containing h_i neurons, the computational complexity of GHNN has $\mathcal{O}(m^2 + ((m(m+1))/2)C)$, with $C = \prod_{i=0}^{L} h_i$. The complexity of DGHNN is $\mathcal{O}(m(m + m_q m_p) + m_p(m + m_q)C)$.

In addition, the complexity of HNN is $\mathcal{O}(m^2 + (m+1)C)$. The NSF and CHNN need to calculate the matrix inverse, there their complexity is $\mathcal{O}(m^3 + m^2 + 2(m+1)C)$ and $\mathcal{O}(m^3 + m^2(1 + 12r) + (m+1)C))$, where r is number of constraint.

In general, the complexity of the above-mentioned methods has a relationship of $\mathcal{O}_{\text{HNN}} < \mathcal{O}_{\text{GHNN}} < \mathcal{O}_{\text{DGHNN}} < \mathcal{O}_{\text{NSF}} < \mathcal{O}_{\text{CHNN}}$. When system has a lot of constraints, i.e., $m_q > n$, there has $\mathcal{O}_{\text{GHNN}} > \mathcal{O}_{\text{DGHNN}}$.

IV. Experiments

A. Experiment Setup and Training Details

1) Simulated Systems: To demonstrate the effectiveness of GHNN, simulations, and experiments on various dynamic systems using different coordinates are conducted. The main evaluation task is long-term trajectory prediction, which is a common benchmark for assessing the accuracy of data-driven models. The GHNN is compared with several existing models as follows.

1) The *NODE* model [34], which served as the black-box baseline.
2) The *HNN* model [14], a well-known classic Hamiltonian-inspired model suitable for canonical coordinates.
3) The *NSF* model[2] [31], a promising Hamiltonian-inspired model designed for general symplectic 2-form coordinates.
4) The *CHNN* model[3] [30], a Hamiltonian-inspired model designed to handle systems with holonomic constraints explicitly.

The following coordinates are considered for simulations.

1) *Canonical Coordinate:* The observation is the canonical position and momenta denotes as $\boldsymbol{z} = [\boldsymbol{q}^{\mathrm{T}}, \boldsymbol{p}^{\mathrm{T}}]^{\mathrm{T}}$. Specifically, for the pendulum system, the canonical position is swing angles $\boldsymbol{q} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^{\mathrm{T}}$ and the canonical momenta is the angular momenta of each body. All five models are compared in this coordinate. It should be mentioned that in this system, the CHNN without constraints is equivalent to the HNN.
2) *Velocity Coordinate:* The observation is the canonical position and velocity $\boldsymbol{y} = [\boldsymbol{q}^{\mathrm{T}}, \dot{\boldsymbol{q}}^{\mathrm{T}}]^{\mathrm{T}}$. This coordinate system is specifically tested on the double pendulum system since the difference between velocity and momenta in a simple pendulum system only has a constant scale factor. In this coordinate, normal Hamiltonian functions struggle to be utilized theoretically. As a result, HNN and CHNN could not be applied.
3) *S^1 Space:* The observation is $\boldsymbol{y} = [\cos \boldsymbol{q}^{\mathrm{T}}, \sin \boldsymbol{q}^{\mathrm{T}}, \boldsymbol{p}^{\mathrm{T}}]^{\mathrm{T}}$. This coordinate treats the general position as an embedded manifold in $\mathbf{R}^{2n}$, which helps avoid singular points and reduce nonlinearity [26]. Since this is neither a canonical nor a symplectic 2-form coordinate, it is not possible to construct HNN and NSF models in this coordinate. Therefore, only the NODE model is compared with GHNN.
4) *Cartesian Coordinate:* The observations are represented by position and linear momenta, as $\boldsymbol{y} = [\boldsymbol{x}^{\mathrm{T}}, \boldsymbol{y}^{\mathrm{T}}, \boldsymbol{p}_x^{\mathrm{T}}, \boldsymbol{p}_y^{\mathrm{T}}]^{\mathrm{T}}$. This coordinate is commonly used because it's relatively easier to measure the position and velocity, and the linear momentum can be easily

[2]The NSF model requires to calculate the inverse of the skew-symmetric matrix, which can often become singular during training when the NN converges to unstable regions. Tuning the hyperparameters can be a challenging task, making practical utilization difficult.

[3]The CHNN model requires the calculation of the inverse of the constraint matrix in (12). However, the authors find that it either suffered from singularity or failed to converge during training. This issue mainly arises form the dependency of the seconded term of (11), $\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{q})\nabla_{\boldsymbol{p}}\hat{\mathcal{H}}(z)$, on the gradient of the learned $\hat{\mathcal{H}}$, which brings complicate calculate graph and relax the holonomic constraints. This results in a complex computational graph and relaxation of the holonomic constraints. Consequently, gradient descent struggles to propagate effectively during training, and (12) tends to encounter singularities when fails to converge to desirable regions. Even after extensive hyperparameters fine-tuning, the authors fail to recover the CHNN models. Therefore, as a workaround, this article assumes the general momenta and velocities are readily available. By doing so, the constraint function of CHNN simplifies to $\boldsymbol{N}^{\mathrm{T}}(\boldsymbol{q})\dot{\boldsymbol{x}}$, which is independent of the learned $\hat{\mathcal{H}}$ and eases the training process of the CHNN.

calculated with items as $p_i = m_i v_i$, where m_i is the mass of the ith object. Since canonical HNN [14] cannot handle the constraints, the NODE, NSF, and CHNN models are compared with GHNN.

2) Training Detail: For all tasks, the simulators integrate the corresponding dynamics for 300-time steps, utilizing a time interval of 0.01 s to generate trajectory data for constructing datasets. Each trajectory begins with random initial conditions. Throughout the training process, all models employ a decay learning rate that initiates at 10^{-3} and utilizes a batch size of $M = 1000$ with the Adam optimizer. Training continues with 5000 gradient steps, which is sufficient for convergence. Subsequently, evaluation is conducted using the same test set.

A fourth-order Runge–Kutta method is employed to solve the learned ODE equation. For GHNN and HNN, the forward training time step has $I = 7$, while NSF and CHNN use $I = 5$. In the case of the NODE model $I = 3$, as lager values of I tend to result in worse performance or even divergence.[4] The random resampled coefficient is sampled from a uniform distribution $c \sim U(1, 10)$. For a simple system, the balance parameter α is set lower or even zero, such as $\alpha = 0$ for a simple pendulum system. For a complex double pendulum system, a higher value $\alpha = 5$ is utilized.

For GHNN, it is recommended that hyperparameters be chosen within specific ranges: $I \in [1, 10]$, $c \in [1, 10]$, and $\alpha \in [0, 5]$. For more complex systems, a lower value of I, c, but a higher value of α is advised, and vice versa, to optimize performance.

3) Evaluating Performance: To evaluate the performance, all the models predict the trajectories starting from the same initial condition for 300 steps. The relative error [30] is used, defined as

$$E = \frac{\|\hat{z} - z\|_2}{\|\hat{z}\|_2 + \|z\|_2}. \tag{38}$$

To evaluate the performance over a trajectory, the geometric mean error is used

$$GE = \exp\left(\frac{1}{N}\sum_{t=i}^{N} \log E_t\right). \tag{39}$$

B. Simple Pendulum System

For the given gravitational constant g, the pendulum length l, and the mass m, the simple pendulum system can be written as the Hamiltonian is

$$\begin{bmatrix} \dot{\alpha} \\ \dot{p}_\alpha \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \nabla \mathcal{H}(\alpha, p_\alpha) \tag{40}$$

where $\mathcal{H} = mgl(1-\cos\alpha) + (p_\alpha^2/2ml^2)$, α is the swing angle, and p_α is the angular momentum of the body. Subsequently, extend this canonical Hamiltonian system to S^1 space

$$\begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -u_2 \\ 0 & 0 & u_1 \\ u_2 & -u_1 & 0 \end{bmatrix} \nabla \bar{\mathcal{H}}(u_1, u_2, v) \tag{41}$$

where $\bar{\mathcal{H}}(\boldsymbol{u}, \boldsymbol{v}) = mgl(1 - u_1) + (v^2/2ml^2)$, and system state has $\boldsymbol{y} = [u_1, u_2, v]^{\mathrm{T}} = [\cos\alpha, \sin\alpha, p_\alpha]^{\mathrm{T}}$.

[4]One possible explanation is that the absence of prior physical laws limits the NN's ability to learn through long forward propagation.

For decomposition $\boldsymbol{J}$, there has

$$\boldsymbol{A} = \begin{bmatrix} -u_2 \\ u_1 \end{bmatrix}, \boldsymbol{B} = 1, \boldsymbol{C} = 0. \tag{42}$$

This coordinate alleviates the nonlinearity of the system.

Furthermore, if considering the pendulum system represented on the Cartesian coordinate, the motion of the system is governed by

$$\begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{u_2^2}{l^2} & -\frac{u_1 u_2}{l^2} \\ 0 & 0 & -\frac{u_1 u_2}{l^2} & \frac{u_1^2}{l^2} \\ -\frac{u_2^2}{l^2} & \frac{u_1 u_2}{l^2} & 0 & \frac{v_1 u_2 - v_2 u_1}{l^2} \\ \frac{u_1 u_2}{l^2} & -\frac{u_1^2}{l^2} & \frac{v_2 u_1 - v_1 u_2}{l^2} & 0 \end{bmatrix} \nabla \bar{\mathcal{H}}(\boldsymbol{u}, \boldsymbol{v}) \tag{43}$$

where $\bar{\mathcal{H}}(\boldsymbol{u}, \boldsymbol{v}) = (1/2m)(v_1^2 + v_2^2)$, and system state has $\boldsymbol{y} = [u_1, u_2, v_1, v_2]^{\mathrm{T}} = [x, y, p_x, p_y]^{\mathrm{T}}$.

For decomposited $\boldsymbol{J}$, there has

$$\boldsymbol{A} = \frac{1}{l}\begin{bmatrix} -u_2 \\ u_1 \end{bmatrix}, \boldsymbol{B} = \frac{1}{l}\begin{bmatrix} -u_2 \\ u_1 \end{bmatrix}, \boldsymbol{C} = \frac{1}{l}\begin{bmatrix} -v_2 \\ v_1 \end{bmatrix}. \tag{44}$$

The state in this coordinate can be directly observed using common sensors, and this coordinate makes the Hamiltonian quantity $\bar{\mathcal{H}}$ very easy to express and learn.

To verify the ability of GHNN for learning above three different coordinates, here generating 1000 trajectories, starting from $\alpha(t_0) \in [-(\pi/4), (\pi/4)]$ with $m = 1$ kg, $l = 1$ m, and $g = 9.8$ m/s^2.

The predictive outcomes of three distinct coordinates using different models are presented in Fig. 5. Both the HNN and NSF models accurately forecast high-energy trajectories in the canonical coordinate. In the case of low-energy trajectories, while these models successfully conserve energy, they demonstrate notable prediction inaccuracies. The NODE model outperforms HNN and NSF in predicting low-energy trajectories. Nonetheless, NODE exhibits some deviation in high-energy trajectory predictions in the canonical coordinate, largely due to its lack of a Hamiltonian structure. Remarkably, the predictions generated by the GHNN and DGHNN match the ground truth perfectly, remaining in the true trajectories after long-term propagation.

In the S^1 coordinate, NODE struggles to account for redundant states and often overlooks minor fluctuation in $\cos\alpha$, basing predictions solely on the evolution of the other two observations. In contrast, GHNN and DGHNN effectively learn the underlying structure of dynamics in this coordinate, enabling stable, long-term predictions. The DGHNN particularly excels in predicting high-energy trajectories, benefiting from the deep decomposition of the $\boldsymbol{J}$ matrix.

In the Cartesian coordinate, NSF fails to capture the dynamics accurately, leading to divergences in long-term predictions. NODE models show energy decay in high-energy trajectories. CHNN achieves stable long-term predictions by incorporating known constraints as priors, albeit with a slight tendency to

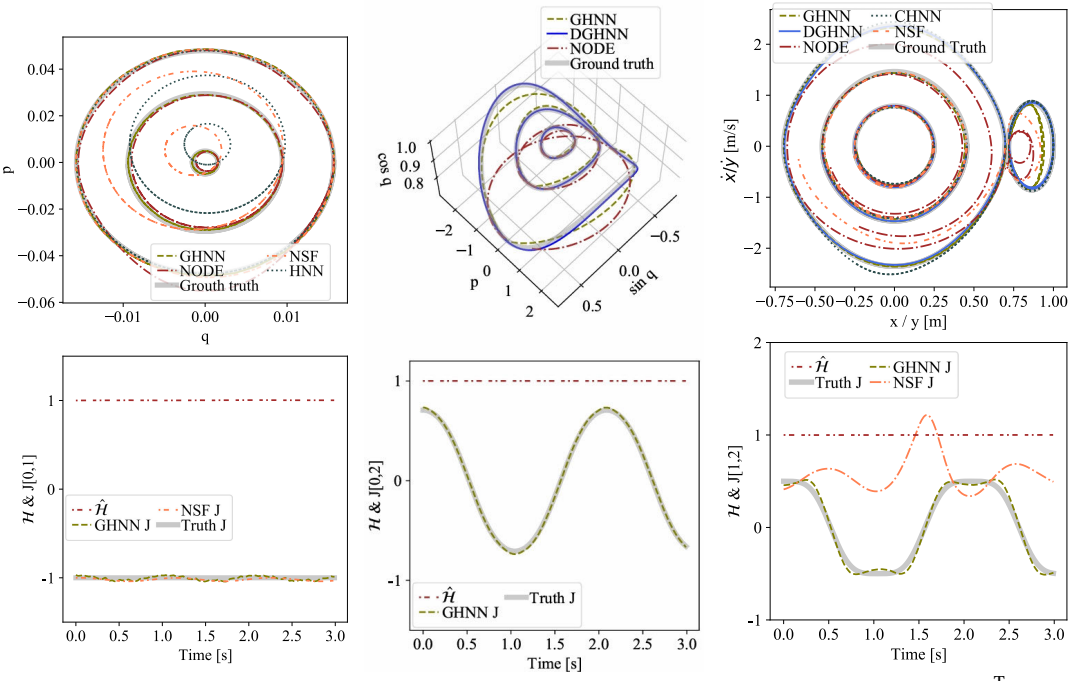


Fig. 5. Performance of different models in various coordinates. First column: results in the canonical coordinate with $\boldsymbol{z} = [\alpha, p]^{\mathrm{T}}$; second column: results under the S^1 coordinate with $\boldsymbol{y} = [\cos\alpha, \sin\alpha, p]^{\mathrm{T}}$; third column: results in the Cartesian coordinate with $\boldsymbol{y} = [x, y, p_x, p_y]^{\mathrm{T}}$. First row: long-term predictions from a certain initial state; second column: $\hat{\mathcal{H}}$ value predicted by HNet and $\hat{\boldsymbol{J}}$ value predicted by JNet, which scaled by $\beta = -0.4, 0.33$, and 0.5 for three different coordinates, respectively.

overestimate high-energy trajectories and underestimate low-energy ones. Meanwhile, GHNN delivers precise long-term predictions without relying on prior constraint knowledge. Notably, both NSF and NODE models perform poorly in the y-direction, where the y state is redundant and changes slightly over time. DGHNN shows promising results, providing precise prediction and learning constraints directly from data through the decomposed $\boldsymbol{J}$. Overall, the predictions made by GHNN and DGHNN achieve better agreement with the ground truth across all coordinates, as evidenced by statistic errors presented in Table II clearly.

Meanwhile, the GHNN is also able to recover the underlying structure of the system, as shown on the second row in Fig. 5. The predicted total energy, denoted as $\hat{\mathcal{H}}$, by GHNN remains constant along a certain trajectory. This indicates that GHNN effectively learns energy conservation across all coordinate systems. In addition, Fig. 5 also displays the scaled element values of predicted $\hat{J}$. NSF accurately predicts $\hat{J}$ in the simple canonical coordinate but encounters difficulty in the Cartesian coordinate, with learned $\hat{J}$ retrieving the underlying structure. This implies that the NSF model lacks interpretability and is unstable. On the contrary, the predicted $\hat{J}$ values of GHNN closely align with the truth value as (3), (7), or (8), verifying that GHNN genuinely captures the functional relationship directly from data. These results confirm that the data-driven dynamic models leaned by GHNN share typical characteristics of analytic models without requiring specific knowledge about individual systems.

C. Double Pendulum System

In the second scenario, the double pendulum system is considered, in which the dynamics become progressively complex and chaotic. Similarly, the dataset is collected by beginning with various initial states, and the test dataset varies from the training.

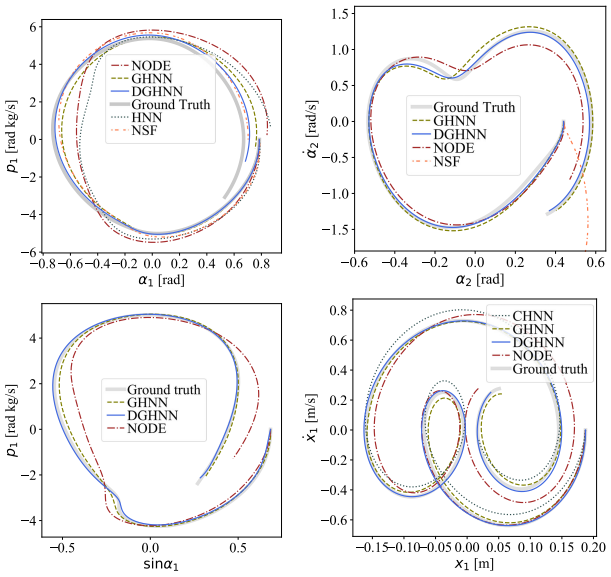


Fig. 6. Performance of different models in double pendulum system. Left up: results in the canonical coordinate with $\boldsymbol{z} = [\alpha_1, \alpha_2, p_1, p_2]^{\mathrm{T}}$; right up: results in the velocity coordinate with $\boldsymbol{y} = [\alpha_1, \alpha_2, \dot{\alpha}_1, \dot{\alpha}_2]^{\mathrm{T}}$; left bottom: results in the S^1 coordinate with $\boldsymbol{y} = [\cos\alpha_1, \sin\alpha_1, \cos\alpha_2, \sin\alpha_2, p_1, p_2]^{\mathrm{T}}$; right bottom: results in the Cartesian coordinate with $\boldsymbol{y} = [x_1, y_1, x_2, y_2, p_{x_1}, p_{y_1}, p_{x_2}, p_{y_2}]^{\mathrm{T}}$.

The prediction results of different models are shown in Fig. 6, where both GHNN and DGHNN perform admirably across all coordinates. In the canonical coordinate, due to the absence of a Hamiltonian structure, NODE experiences

a quick decline in precision over time. The prediction errors of HNN and NSF also incrementally rise and are notably inferior compared to the proposed methods. DGHNN delivers the best-performing prediction in this coordinate. In velocity coordinates, the NSF model faces rapid divergence, attributed to the complexity of training with a high-dimensional inverse matrix. Conversely, GHNN and DGHNN manage to provide accurate long-term prediction. Similar results are also observed in the S^1 coordinate, where GHNN and DGHNN shine, while NODE struggles with significant prediction inaccuracies. Despite explicitly incorporating constraints, CHNN encounters substantial errors in the Cartesian coordinate due to its intricate computational framework. GHNN maintains a low error margin in this coordinate, whereas NODE experiences considerable divergence. DGHNN achieves the best performance in this chaotic system through the decomposition of the interconnection matrix, effectively mastering constraint laws directly from the data.

Overall, GHNN and DGHNN consistently prove their exceptional capability to accurately model complex dynamics across different coordinates. This assertion is conclusively backed by statistical evidence presented in Table II. Furthermore, the training of GHNN without accounting for energy loss results in prediction errors exceeding 100 across various coordinates, surpassing even those of the NODE method. These findings underscore the critical importance of incorporating energy loss to enhance the effectiveness of the HNet.

D. Coupled Pendulum System

In the third scenario, GHNN is tested using a coupled pendulum system to evaluate its capacity to model constrained dynamic systems, as shown in Fig. 3. It's important to note that deriving the generalized momenta for such systems is a significant challenge. As a result, the two different coordinates considered for the coupled system both involve position-velocity pairs. In the first coordinate, the observations consist of angles and angular velocities, denoted as $\boldsymbol{y} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \dot{\alpha}_1, \dot{\alpha}_2, \dot{\alpha}_3, \dot{\alpha}_4]^{\mathrm{T}}$. In the second coordinate, which utilizes Cartesian coordinates, observations contain the positions and linear velocities, represented as $\boldsymbol{y} = [x_1, y_1, x_2, y_2, \dot{x}_1, \dot{y}_1, \dot{x}_2, \dot{y}_2]^{\mathrm{T}}$.

Due to the absence of general momenta in the angular coordinate, CHNN is not applicable in this condition. While momenta can be determined in the Cartesian coordinate, CHNN fails to converge.

The prediction results are depicted in Fig. 7. Notably, both GHNN and DGHNN maintain minimal prediction errors in modeling the complex coupled dynamics. In contrast, the NODE model exhibits increasing prediction errors and diverges in long-term predictions. In the angular coordinate, GHNN shows a minor energy decay, whereas DGHNN perfectly covers the ground truth trajectory. In Cartesian coordinates, DGHNN notably predicts the complex evolution in the y-direction with high precision, while GHNN also demonstrates minor prediction errors. These findings confirm that GHNN and DGHNN learn the constraints from data automatically, with DGHNN exhibiting superior performance. The

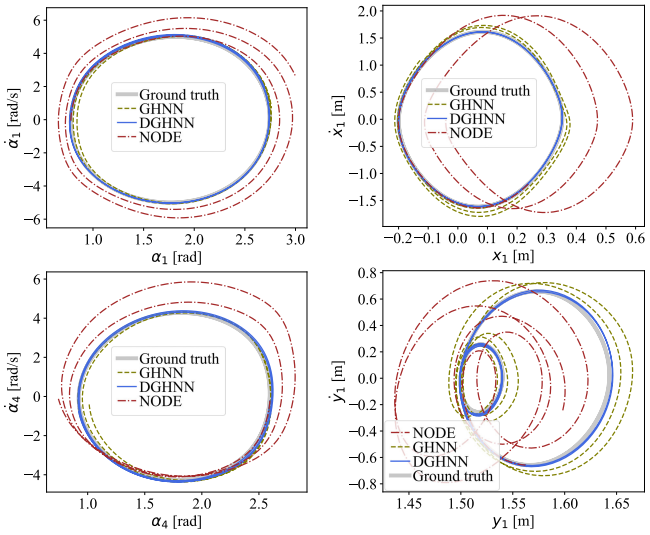


Fig. 7. Performance of different models in the coupled system. Left: results in the angle coordinate; right: results in the Cartesian coordinate.

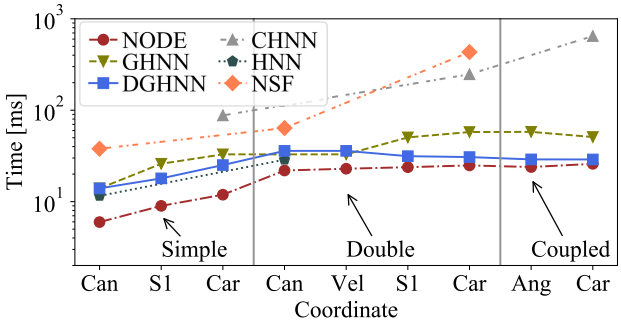


Fig. 8. Prediction time of different models. Can: the canonical coordinate; S1: the S^1 coordinate; Car: the Cartesian coordinate; Vel: the velocity coordinate.

impressive results of DGHNN in handling constrained systems verify the effectiveness of the decomposed $\boldsymbol{J}$ framework.

In summary, Table II provides a comprehensive comparison of all results. The black-box NODE model, while capable of learning the dynamics of various systems, often results in higher prediction errors. Conversely, models like HNN, NSF, and CHNN, which are designed with specific features in mind, generally improve performance in their respective conditions. However, their applicability is somewhat limited, and they may even diverge in certain test trajectories.[5] Overall, the proposed methods outperform all other models in handling various systems considered in this article. This demonstrates the proposed method's capability to model complex dynamics in arbitrary coordinates and learn constraints automatically.

Across all test cases, GHNN and DGHNN demonstrate superior speeds compared to most Hamiltonian-inspired models, with performance nearing that of normal NNs (NODE), as illustrated in Fig. 8. The specific computational graph

[5]For noncanonical coordinates, the NSF and CHNN models rely on the inner-connection matrix $\boldsymbol{J}$ and its inverse. When the eigenvalues of this matrix are extremely to zeros ($\epsilon \ll 1$), the model's rollout and the optimization of the forward model can become numerically unstable. This highlights that avoiding explicit inversion contributes to greater stability in both learning and model rollout.

TABLE II

TEST GE FOR DIFFERENT DYNAMIC SYSTEMS. THE RESULTS IS MEASURED BY GE AND MULTIPLIED BY 10^3 FOR ALL SYSTEMS AND THE BEST ARE EMPHASIZED BY BOLD

	NODE	HNN	NSF	CHNN	GHNN	DGHNN
Simple pendulum (canonical)	130.78 ± 109.31	16.75 ± 39.04	51.98 ± 39.44	as HNN	**11.10 ± 4.20**	as GHNN
Simple pendulum (S^1)	45.05 ± 54.95	-	-	-	**5.98 ± 2.10**	8.12 ± 6.97
Simple pendulum (Cartesian)	62.24 ± 63.76	-	39.36 ± 46.92	32.49 ± 41.77	8.73 ± 2.44	**7.29 ± 4.92**
Double pendulum (canonical)	82.36 ± 62.00	24.04 ± 33.59	22.21 ± 26.63	as HNN	20.13 ± 17.81	**18.87 ± 16.26**
Double pendulum (velocity)	125.66 ± 73.55	-	288.26 ± 122.96	-	49.24 ± 34.02	**32.78 ± 20.99**
Double pendulum (S^1)	30.82 ± 24.66	-	-	-	**8.25 ± 4.55**	19.12 ± 17.25
Double pendulum (Cartesian)	98.23 ± 82.47	-	× [6]	41.96 ± 22.81	22.01 ± 19.47	**18.11 ± 8.85**
Couple pendulum (angular)	139.15 ± 20.68	-	-	-	27.46 ± 8.06	**16.75 ± 6.91**
Couple pendulum (Cartesian)	99.47 ± 22.78	-	-	×	30.47 ± 6.37	**9.41 ± 5.47**

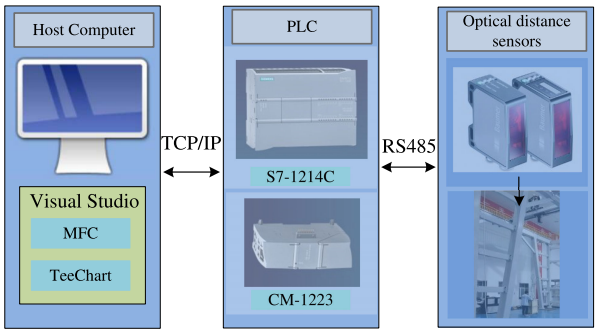


Fig. 9. Three-dimensional crane experiment setup.

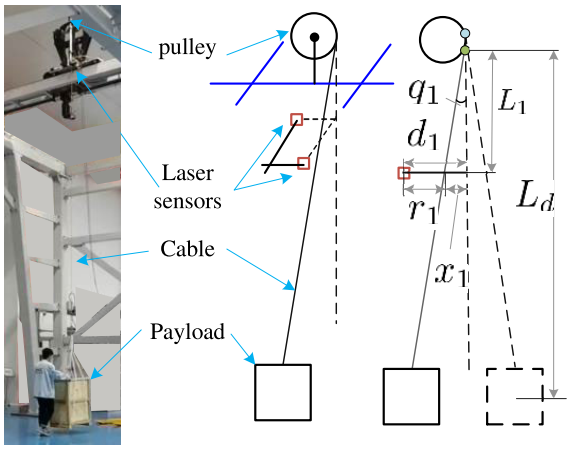


Fig. 10. Measurement configuration in the 3-D crane.

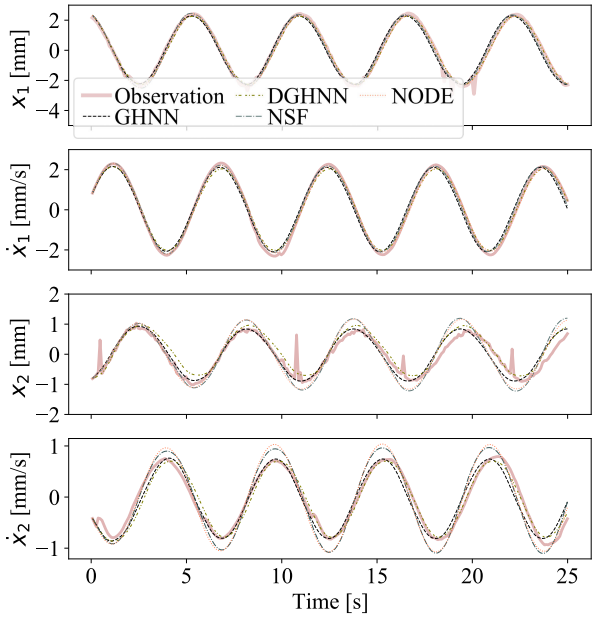


Fig. 11. Prediction results in the 3-D crane system.

designs tailored for the respective coordinates result in NSF and GHNN exhibiting longer prediction times, which increase rapidly as the state dimension enlarges. DGHNN exhibits a marginally higher computational cost due to matrix multiplication but achieves faster performance in systems with constraints, which is consistent with the analysis provided in Section III-E.

E. Real 3-D Crane

Next, the GHNN model is applied to a real 3-D overhead crane system. The 3-D crane system is established using a PLC/optical sensor framework, as depicted in Fig. 9. The system consists of a payload with mass m suspended by a cable with length L_d passing through a pulley, as shown in Fig. 10. Two laser sensors are set orthogonally to measure the distance between the senor with cable, providing observations x_1, x_2, which have

$$x_1 = L(q_1, q_2)\tan q_1, \quad x_2 = L(q_2, q_2)\tan q_2 \tag{45}$$

where q_1 and q_2 represent the swing angles of the payload, and L denotes the distance between the set position of sensors and the pulley's suspension position, which varies as the cable swinging. The observations are nonlinear mappings of the canonical states q_1 and q_2.

[6]The NSF fails to converge, leading to the singularity of the learned matrix J, when attempting to learn double pendulum models under Cartesian coordinate. This may be attributed to the high state dimension in this condition, as the $\boldsymbol{J} \in \mathbf{R}^{8\times 8}$ for Cartesian coordinate on the double pendulum system has a high dimension.

In this practical system, only the implicit position is directly observed, and the velocities $\dot{x}_1, \dot{x}_2$ are calculated using the tracking differentiator [35]. Consequently, the standard HNN and CHNN models are not applicable because the momenta is

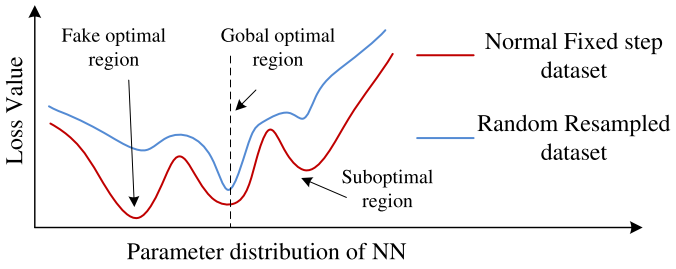


Fig. 12. Loss on different datasets.

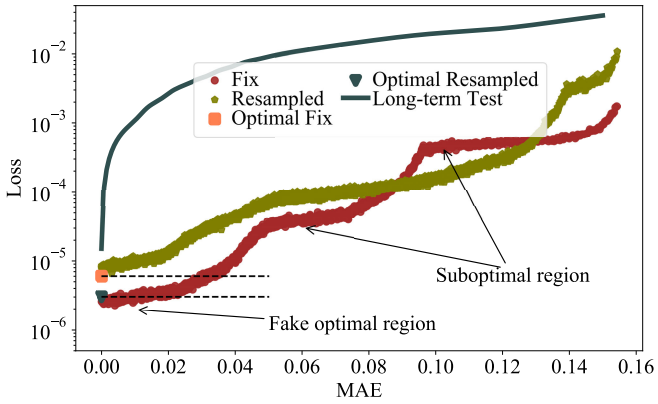


Fig. 13. Loss on different datasets for a linear differential equation.

not observed. These velocities are mappings of the canonical momenta

$$\dot{x}_1 = \frac{L_1(q_1)p_1}{L_d m}, \quad \dot{x}_2 = \frac{L_2(q_2)p_2}{L_d m}. \tag{46}$$

Due to practical sensor configurations, this 3-D crane system is described under a local coordinate with observation $z = [x_1, x_2, \dot{x}_1, \dot{x}_2]^{\mathrm{T}}$. Equations (45) and (46) reveal that this local coordinate is a mapping of the canonical coordinate, allowing the GHNN to learn. In addition, in this special coordinate, the observation is two-form symplectic. Consequently, only NSF and NODE models can applied, and are compared with GHNN and DGHNN.

The dataset is collected during a free swing of the system, lasting for 120 s, with a sampling frequency of 10 Hz.

As depicted in Fig. 11, all models perform well in the x_1 direction, with the predicted trajectory closely matching the ground truth. However, for the x_2 direction, the prediction error of NODE and NSF models exceeds 0.5 mm after 20 s. These substantial amplitude overestimations may be attributed to the high noise level in the sensors measuring the x_2 direction. In contrast, GHNN and DGHNN demonstrate precise predictions, maintaining the accuracy of up to 25 s of forward propagation in both two directions. These results highlight the robustness of the proposed method in handling highly noisy practical systems, and its ability to effectively capture dynamics in local coordinate systems which are limited by measurement equipment.

V. Conclusion

This article proposes a versatile and generalized Hamiltonian framework, integrating with NNs to establish a GHNN model. GHNN is designed to capture dynamics on arbitrary coordinates and automatically handle complex constraints. This model consists of two individual NNs, JNet, and HNet. HNet predicts the Hamiltonian quantity and its gradient is further calculated and multiplied by the interconnection matrix predicted by JNet to generate state deviations. The prediction of the coming states is obtained through an ODE solver. Through in-depth comparisons with baseline black-box NODE models and other state-of-the-art Hamiltonian-inspired models, GHNN demonstrates remarkable performance. Experiment and simulation results confirm that the proposed method can provide accurate prediction for different coordinate systems and shows robustness in practical application. These results strongly indicate that GHNN is not system- or coordinate-specific and can effectively learn dynamics on various reasonable coordinates. Collectively, these results provide a solid foundation for advancing the field of complex systems modeling.

Although this article primarily focuses on the autonomous (uncontrolled) setting, it's important to note that the method is highly compatible with controlled systems and opens up several promising directions for future work. These include integration into dynamics systems for model-based control or reinforcement learning settings [8], [26], [36], and extending its applicability to systems involving dissipative forces [37]. Future works will incorporate a broader class of physics-based prior, such as the port-Hamiltonian system formulation. Furthermore, the next step will be to design energy-shaping control strategies based on these interpretable data-driven frameworks, offering valuable insights for control in complex systems.

Appendix

Analyze of Random Resample Method

Supposing the ODEsolver used by learning models to approximate the differential equation is the Euler method, which has

$$\boldsymbol{y}_{t+1} = f(\boldsymbol{y}_t) \cdot h + \boldsymbol{y}_t + T(\boldsymbol{y}) \simeq f(\boldsymbol{y}) \cdot h + \boldsymbol{y}_t \tag{47}$$

where $T(\boldsymbol{y})$ is the truncation error. For the learning process, the loss function is the distance between the prediction state with the ground truth, thus the data-driven model eventually learns

$$\bar{\boldsymbol{f}}(\boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{y}) + T(\boldsymbol{y})/h \tag{48}$$

but not $\boldsymbol{f}(\boldsymbol{y})$ itself. Thus, this inducts a truncation error basis for data-driven models and may bring a lot of suboptimal regions causing difficulty in converging. Worse still, there possibly be a "Fake optimal region" where models show minimized training loss but the corresponding parameter betrays the underlying structure of the system as shown in Fig. 12.

On the proposed resampled dataset, the next state has

$$\boldsymbol{y}_{t+1} = \frac{1}{M} \sum_{i=1}^{M} \left[\boldsymbol{f}(\boldsymbol{y}_{t-i}) \cdot ih + \boldsymbol{y}_{t-i} + T_i(\boldsymbol{y}_{t-i}) \right]. \tag{49}$$

This makes the data-driven model learn

$$\bar{\boldsymbol{f}}_{\mathrm{rs}}(\boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{y}) + \frac{1}{M}\sum_{i=1}^{M}\left(T_i(\boldsymbol{y}_{t-i})\right). \tag{50}$$

Conjecture 1: For a sufficient number of resampled data, the truncation error's expectations of the ODEsolver approximates to zero, which has

$$\mathbb{E}(T_i(\boldsymbol{y}_{t-i})) = 0. \tag{51}$$

Thus, the truncation error on this resampled dataset can be viewed as the aleatoric uncertainty, which substantially diminishes the number of potential fake and suboptimal solutions.

To evaluate Conjecture 1, a simulation involving a simple linear differential equation system is conducted. The system's dynamics are governed by

$$\dot{\boldsymbol{y}} = \boldsymbol{W}\boldsymbol{y} \tag{52}$$

where $\boldsymbol{W} \in \mathbf{R}^{5\times 5}$ is a constant matrix. By introducing various disturbances to the matrix $\boldsymbol{W}$, the approximated $\hat{\boldsymbol{W}}$ can be derived. Subsequently, the resulting loss corresponding to the accuracy of $\hat{\boldsymbol{W}}$ across different datasets is illustrated in Fig. 13. This setup allows one to explore how well the learning process can approximate $\boldsymbol{W}$ using different training sets.

The accuracy of the approximated $\hat{\boldsymbol{W}}$, compared to the true matrix $\boldsymbol{W}$, is measured using the means absolute error (MAE) defined as

$$\mathrm{MAE} = \sum_{i=1}^{5}\sum_{j=1}^{5}|\hat{\boldsymbol{W}}_{ij}| - |\boldsymbol{W}_{ij}|. \tag{53}$$

The results indicate a smoother decrease in loss on the resampled dataset, highlighting the benefits of the random resampled method in improving model generalization. In contrast, the loss associated with the fixed step dataset exhibits numerous suboptimal regions and even fake optimal regions. This behavior suggests that, when the learned matrix approaches the ground truth $\boldsymbol{W}$, the model might incorrectly identify a solution as optimal. These findings support the idea that the random resampled method can significantly enhance the learning process, mitigate the risk of settling on suboptimal solutions, and foster a more accurate approximation of the underlying dynamics.

Less accurate integrators, especially nonsymplectic ones, introduce amounts of truncation error, which can greatly amplify divergence in chaotic trajectories. Symplectic integrators have been demonstrated to enhance prediction stability for Hamiltonian models. Furthermore, they may mitigate the impact of truncation errors during training, which will be discussed in future work.

REFERENCES

[1] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Trans. Robot.*, vol. 33, no. 6, pp. 1292–1312, Dec. 2017.

[2] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Proc. Conf. Robot Learn.*, 2020, pp. 1–10.

[3] A. D. Libera and R. Carli, "A data-efficient geometrically inspired polynomial kernel for robot inverse dynamic," *IEEE Robot. Autom. Lett.*, vol. 5, no. 1, pp. 24–31, Jan. 2020.

[4] M. O'Connell et al., "Neural-fly enables rapid learning for agile flight in strong winds," *Sci. Robot.*, vol. 7, no. 66, pp. 65–97, May 2022.

[5] R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke, "Explainable machine learning for scientific insights and discoveries," *IEEE Access*, vol. 8, pp. 42200–42216, 2020.

[6] G. Carleo et al., "Machine learning and the physical sciences," *Rev. Mod. Phys.*, vol. 91, pp. 45–52, Dec. 2019.

[7] M. Wang, X. Lou, W. Wu, and B. Cui, "Koopman-based MPC with learned dynamics: Hierarchical neural network approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 8, no. 3, pp. 3896–3906, Oct. 2022.

[8] M. Lutter, C. Ritter, and J. Peters, "Deep Lagrangian networks: Using physics as model prior for deep learning," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[9] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, "Discovering physical concepts with neural networks," *Phys. Rev. Lett.*, vol. 124, no. 1, Jan. 2020, Art. no. 010508.

[10] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, Apr. 2009.

[11] M. Lutter and J. Peters, "Combining physics and deep learning to learn continuous-time dynamics models," *Int. J. Robot. Res.*, vol. 42, no. 3, pp. 83–107, 2023.

[12] A. R. Geist and S. Trimpe, "Structured learning of rigid-body dynamics: A survey and unified view from a robotics perspective," *GAMM-Mitteilungen*, vol. 44, no. 2, 2021, Art. no. e202100009.

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

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

[15] J. Sun, Z. Ma, Y. Tian, and M. Qin, "Symplectic structure of Poisson system," *Appl. Math. Mech.*, vol. 26, no. 11, pp. 1345–1350, 2005.

[16] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou, "Symplectic recurrent neural networks," in *Proc. Int. Conf. Learn. Represent.*, Apr. 2020, pp. 1–23.

[17] R. Bondesan and A. Lamacraft, "Learning symmetries of classical integrable systems," in *Proc. ICML Workshop Theor. Phys. Deep Learn.*, Jun. 2019, pp. 1–8.

[18] S.-H. Li, C.-X. Dong, L. Zhang, and L. Wang, "Neural canonical transformation with symplectic flows," *Phys. Rev. X*, vol. 10, no. 2, Apr. 2020, Art. no. 021020.

[19] D. M. DiPietro, S. Xiong, and B. Zhu, "Sparse symplectically integrated neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 6074–6085.

[20] S. Xiong, Y. Tong, X. He, S. Yang, C. Yang, and B. Zhu, "Nonseparable symplectic neural networks," 2020, *arXiv:2010.12636*.

[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] Y. Tong, S. Xiong, X. He, G. Pan, and B. Zhu, "Symplectic neural networks in Taylor series form for Hamiltonian systems," *J. Comput. Phys.*, vol. 437, Jul. 2021, Art. no. 110325.

[23] C. L. Galimberti, L. Furieri, L. Xu, and G. Ferrari-Trecate, "Hamiltonian deep neural networks guaranteeing nonvanishing gradients by design," *IEEE Trans. Autom. Control*, vol. 68, no. 5, pp. 3155–3162, May 2023.

[24] P. Jin, Z. Zhang, I. G. Kevrekidis, and G. E. Karniadakis, "Learning Poisson systems and trajectories of autonomous systems via Poisson neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 8271–8283, May 2023.

[25] M. Šípka, M. Pavelka, O. Esen, and M. Grmela, "Direct Poisson neural networks: Learning non-symplectic mechanical systems," *J. Phys. A, Math. Theor.*, vol. 56, no. 49, Nov. 2023, Art. no. 495201.

[26] Y. Zhong, B. Dey, and A. Chakraborty, "Symplectic ODE-Net: Learning Hamiltonian dynamics with control," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–16.

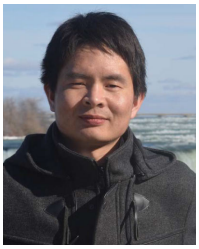
[27] T. Duong and N. Atanasov, "Hamiltonian-based neural ODE networks on the SE(3) manifold for dynamics learning and control," in *Proc. RSS*, Jul. 2021, pp. 1–13.

[28] R. Rashad, F. Califano, and S. Stramigioli, "Port-Hamiltonian passivity-based control on SE(3) of a fully actuated UAV for aerial physical interaction near-hovering," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4378–4385, Oct. 2019.

[29] Y. Okura, K. Fujimoto, and C. Kojima, "Virtual holonomic constraints control for port-Hamiltonian systems: A case study of fully actuated mechanical systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5598–5603, 2020.

[30] M. Finzi, K. A. Wang, and A. G. Wilson, "Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints," in *Proc. 34th Conf. Neural Inf. Process. Syst.*, 2020, pp. 13880–13889.

[31] T. M. Yuhan Chen and T. Yaguchi, "Neural symplectic form: Learning Hamiltonian equations on general coordinate systems," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 16659–16670.
[32] Z. Chen, M. Feng, J. Yan, and H. Zha, "Learning neural Hamiltonian dynamics: A methodological overview," 2022, *arXiv:2203.00128*.
[33] J. Hong, "Structure-preserving numerical methods for stochastic Poisson systems," *Commun. Comput. Phys.*, vol. 29, no. 3, pp. 802–830, Jun. 2021.
[34] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3276–3285.
[35] H. Yang, L. Cheng, J. Zhang, and Y. Xia, "Leader–follower trajectory control for quadrotors via tracking differentiators and disturbance observers," *IEEE Trans. Syst. Man Cybern., Syst.*, vol. 51, no. 1, pp. 601–609, Jul. 2021.
[36] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, Feb. 2015.
[37] Y. Desmond Zhong, B. Dey, and A. Chakraborty, "Dissipative SymODEN: Encoding Hamiltonian dynamics with dissipation and control into deep learning," 2020, *arXiv:2002.08860*.



Yanfang Liu received the B.Sc. degree in aircraft design and engineering from Harbin Engineering University, Harbin, China, in 2008, and the Ph.D. degree in aeronautical and astronautical science and technology from the Harbin Institute of Technology, Harbin, in 2014.

Since 2014, he has been with the Harbin Institute of Technology, where he is currently a Professor with the Department of Aerospace Engineering. His research interests include dynamics modeling, smart materials and structures, vibration isolation, intelligent unmanned systems, and visual inspection with AI.

Xu Wang received the M.S. degree from the Department of Aerospace Engineering, Harbin Institute of Technology, Harbin, China, in 2022, where he is currently pursuing the Ph.D. degree.

His current research interests include space robot control, data-driven dynamics models, and reinforcement learning.

Yituo Song received the B.S. degree from the Department of Aerospace Engineering, Harbin Institute of Technology, Harbin, China, in 2023, where he is currently pursuing the Ph.D. degree.

His current research interests include robot control, data-driven dynamics models, and reinforcement learning

Bo Wang received the B.Sc. degree in information and computing science from Beihang University, Beijing, China, in 2010.

Since 2014, he has been with the Astronaut Research and Training Center of China, Beijing, where he is currently an Assistant Professor. His research interests include human-system integration, human factors evaluation, and intelligent computing.

Desong Du (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in aeronautical and astronautical science and technology from the School of Astronautics, Harbin Institute of Technology, Harbin, China, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree.

From 2021 to 2022, he was a Visiting Ph.D. Student with the Department of Cognitive Robotics, Delft University of Technology, Delft, The Netherlands. His research interests include smart materials and structures, space robots, and artificial intelligence.