Journal of
Marine Science and Engineering

Article

Physics-Informed Dynamics Modeling: Accurate Long-Term Prediction of Underwater Vehicles with Hamiltonian Neural ODEs

Xiang Jin [1,*], Zeyu Lyu [2], Jiayi Liu [3] and Yu Lu [1]

[1] School of Naval Architecture and Ocean Engineering, Dalian Maritime University, Dalian 116026, China
[2] Navigation Mark Office of Hongkong-Zhuhai-Macao Bridge, South China See Navigation Support Center, Ministry of Transport, Zhuhai 519080, China
[3] School of Marine Science, Sun Yat-sen University, Zhuhai 519080, China
* Correspondence: mjinxiang@outlook.com

Abstract

Accurately predicting the long-term behavior of complex dynamical systems is a central challenge for safety-critical applications like autonomous navigation. Mechanistic models are often brittle, relying on difficult-to-measure parameters, while standard deep learning models are black boxes that fail to generalize, producing physically inconsistent predictions. Here, we introduce a physics-informed framework that learns the continuous-time dynamics of an Autonomous Underwater Vehicle (AUV) by discovering its underlying energy landscape. We embed the structure of Port-Hamiltonian mechanics into a neural ordinary differential equation (NODE) architecture, learning not to imitate trajectories but rather to identify the system's Hamiltonian and its constituent physical matrices from observational data. Geometric consistency is enforced by representing rotational dynamics on the SE(3) manifold, preventing numerical error accumulation. Experimental validation reveals a stark performance divide. While a state-of-the-art black-box model matches our accuracy in simple, interpolative maneuvers, its predictions fail catastrophically under complex controls. Quantitatively, our physics-informed model maintained a mean 10 s position error of a mere 3.3 cm, whereas the black-box model's error diverged to 5.4 m—an over 160-fold performance gap. This work establishes that the key to robust, generalizable models lies not in bigger data or deeper networks but in the principled integration of physical laws, providing a clear path to overcoming the brittleness of black-box models in critical engineering simulations.

Keywords: physics-informed machine learning; Hamiltonian neural networks; underwater vehicle dynamics; neural ordinary differential equations; long-term prediction

check for updates

Academic Editor: Weicheng Cui

Received: 22 August 2025
Revised: 23 September 2025
Accepted: 28 September 2025
Published: 3 November 2025

Citation: Jin, X.; Lyu, Z.; Liu, J.; Lu, Y. Physics-Informed Dynamics Modeling: Accurate Long-Term Prediction of Underwater Vehicles with Hamiltonian Neural ODEs. *J. Mar. Sci. Eng.* **2025**, *13*, 2091. https://doi.org/10.3390/jmse13112091

1. Introduction

Autonomous Underwater Vehicles (AUVs) are unmanned ocean observation platforms with intelligence. They are known for their compact size, high maneuverability, and ability to operate in deep-sea environments. AUVs incorporate various sensors enabling them to carry out thorough surveys and research in underwater environments, which allows them to undertake submersible exploration without temporal and spatial constraints.

In marine science, AUVs collect oceanic environmental data, survey seafloor geology and topography, and prospect for resources like oil. In marine engineering, AUVs evaluate dam structures, help maintain underwater foundations, aid divers with their tasks, and perform underwater target observation, search, and rescue missions. In military applications, AUVs detect targets, collect intelligence, perform surveillance and reconnaissance, and engage in anti-submarine warfare and related activities.

Differential equations are typically used to describe the motion model of an underwater vehicle. For a particular type of underwater vehicle, Figure 1 illustrates how certain parameters in the motion equations, like the inertial matrix and the Coriolis–centripetal moment matrix, among others, can be accurately obtained through experimental measurements or theoretical calculations once the interior structure and exterior design are established. Nevertheless, another set of parameters, such as hydrodynamic damping coefficients, require acquisition through computational fluid dynamics (CFD) simulations or water tank experiments, despite the fact that the resulting values frequently display a certain degree of discrepancy from the actual ones [1].

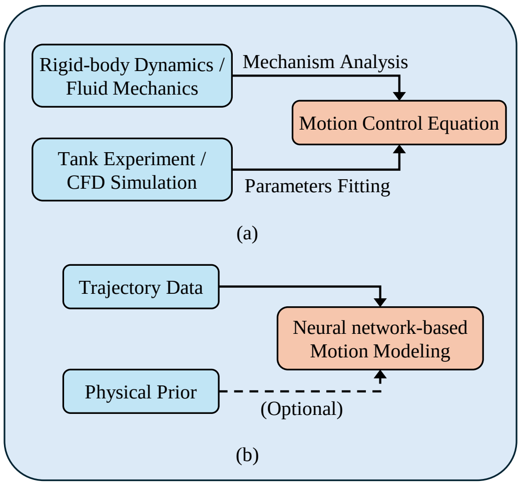

Rigid-body Dynamics / Fluid Mechanics
Mechanism Analysis
Motion Control Equation
Tank Experiment / CFD Simulation
Parameters Fitting
(a)
Trajectory Data
Neural network-based Motion Modeling
Physical Prior
(Optional)
(b)

Figure 1. Underwater vehicle motion modeling methods. (**a**) Traditional modeling method based on mechanism analysis. (**b**) Data-driven modeling method.

Considerable research has focused on overcoming the challenge of estimating inaccurate hydrodynamic parameters. System identification methods have emerged as a common pathway for obtaining high-precision hydrodynamic parameters among various approaches. This method enables a cost-effective online or offline identification of parameters. To identify and optimize parameters, the fundamental approach involves designing parameter identification experiments and subsequently utilizing techniques such as the least squares method [2], Kalman filtering [3], Gaussian processes [4], or neural networks [5].

System identification can attain the desired accuracy by minimizing the discrepancy between theoretical models and actual data [6]. Nevertheless, these methods have some limitations. Firstly, the accuracy of system identification methods depends on the quality of the theoretical model used. In situations where the model is faulty, accurate motion models can still be hard to achieve, even if the identification method shows high precision. Secondly, to completely excite the dynamic characteristics of underwater vehicles, tailored parameter identification experiments must be designed. This leads to a considerable need for high-quality trajectory data. Lastly, when multiple parameters need identification, it may cause challenges for algorithm convergence. It becomes necessary to decouple the motion of the underwater vehicle followed by identifying hydrodynamic parameters in separate channels. Nevertheless, this method may result in inaccuracies when estimating strongly coupled parameters.

Moreover, the theoretical model may lack accuracy inherently because of the absence of a universally accepted method for modeling hydrodynamic damping terms. Numerous methods have been summarized in reference [7] with varying assumptions made regarding the vehicle and fluid characteristics. The generated models differ in their applicability and computational complexity across various scenarios. Furthermore, certain factors within the vehicle dynamics model, including manufacturing errors, assembly discrepancies, external disturbances, and changes in physical characteristics over time, present challenges for explicit modeling. Therefore, predicting the motion behavior of underwater vehicles through differential equations to obtain long-term accurate explanations is exceedingly difficult. Establishing a motion model through experiments, even with a lot of effort, is only applicable to a specific type of vehicle and is difficult to generalize to other vehicles.

Recently, deep learning technology has seen rapid development and has been extensively applied in various fields, including visual recognition [8], natural language understanding [9], robot manipulation, and control [10]. Deep learning has demonstrated much better performance than traditional methods. Deep neural networks are theoretically capable of fitting any nonlinear function, as they serve as universal function approximators. Deep neural networks can automatically extract latent features from large amounts of data, which is supported by carefully designed network architectures and advanced training strategies.

Modeling underwater vehicle dynamics through theoretical analysis presents challenges due to the inherent incompleteness of theoretical models and difficulties associated with obtaining certain parameters accurately. In contrast, deep learning methods can effectively utilize the impressive fitting capacity of neural networks. These methods learn the dynamics of underwater vehicles directly from their trajectory data, surpassing the mere task of fitting parameters within theoretical models.

It is important to note that deep neural networks are black-box models and lack interpretability in their operational mechanisms. Although deep neural networks usually have high model capacity to fit training data well, they may not capture the true underlying patterns in some cases. Instead, they may identify specific local optimal solutions. When confronted with scenarios outside the training data, such models can exhibit significant performance degradation, which is commonly referred to as over-fitting.

To enhance the generalization ability of deep neural networks in unfamiliar contexts and to guide their learning of appropriate data patterns, it is necessary to incorporate task-relevant prior information, i.e., biases, into the learning process. The embedding of bias information into neural networks can be categorized into the following three approaches: a. Input bias: During the data pre-processing stage, domain knowledge can be used to normalize the data by bringing different scales of data to the same. If the data distribution is unfavorable, domain knowledge can be applied to adjust the data distribution, thus facilitating a more effective modeling of the target area [11]. b. Inductive bias: Designing specialized neural network architectures for specific tasks is one way to implicitly embed prior information into the learning process. For example, convolutional neural networks exploit the symmetry and distributed pattern representation present in natural images. By using network structures based on local connectivity and parameter sharing, they have revolutionized the field of computer vision [12–15]. c. Learning bias: Incorporating task-relevant prior information into the loss function as penalty terms is another strategy. This is similar to multi-task learning, where the learning algorithm not only adapts to the training data but also ensures that the network predictions satisfy certain constraints (e.g., conservation of quality and momentum, monotonicity, etc.). Typical methods include deep Galerkin methods [16] and physics-informed neural networks [17,18].

In addition, traditional neural network methods often deal only with discrete data, while the sampling intervals of actual motion data may be irregular. The challenge of recovering continuous dynamics of underwater vehicles from such observational data remains another critical issue. In this study, we establish the neural ordinary differential equation (neural ODE) [19] as a fundamental framework to modeling underwater vehicle dynamics. This framework combines ordinary differential equations with neural networks, starting only from the initial state of the system and using non-uniform observation data to model continuous dynamics. We then express the motion model of underwater vehicles in the form of Hamiltonian mechanics, using the Hamiltonian neural network (HNN) to model system conservation and dissipation quantities, respectively [20,21]. By using exponential mapping methods to handle rotational motion, we ensure that the resulting motion model satisfies the dynamic constraints of underwater vehicles. Finally, we compare the derived dynamic model to a fully data-driven modeling method. The experimental results show that our proposed modeling approach has significant advantages in terms of model robustness and long-term prediction accuracy.

The key contributions of this work are therefore threefold:

- We propose a novel physics-informed framework for AUV dynamics modeling that integrates a Port-Hamiltonian structure within a neural ordinary differential equation (NODE) to explicitly separate and learn energy-conserving and dissipative effects from data.
- We ensure geometric consistency and numerical stability for long-term prediction by representing the full 6-DOF rigid-body motion on the SE(3) manifold, which inherently respects the constraints of rotational dynamics and avoids singularities.
- We provide rigorous quantitative evidence demonstrating that our physics-informed approach overcomes the inherent brittleness of black-box models, maintaining high-fidelity predictions in complex, out-of-distribution scenarios where purely data-driven methods fail.

The paper is organized as follows. Section 2 provides some preliminaries of this work. Section 3 describes the proposed method for the dynamic modeling of underwater vehicles. Experimental results and the discussion are presented in Section 4. And the conclusion is given in Section 5.

2. Sequence Modeling Based on Neural Ordinary Differential Equations

2.1. Problem Statement

We can conceptualize the motion modeling of underwater vehicles as a time series prediction problem. Let us assume that at time t, the motion state of the underwater vehicle can be represented by a vector $\boldsymbol{x}_t$ containing both position and velocity information. Therefore, within the time interval from $t = 0$ to $t = T$, the changing motion states form a sequence $\boldsymbol{X} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$. For simplicity, we assume that the control inputs of the underwater vehicle remain constant during this period. As a result, the motion modeling problem for the underwater vehicle revolves around predicting a sequence of motion states $\boldsymbol{Y} = (\boldsymbol{x}_{T+1}, \boldsymbol{x}_{T+2}, \ldots, \boldsymbol{x}_{T+N})$ for a duration from time $T+1$ to $T+N$ based on the known sequence $\boldsymbol{X}$.

In the context of deep learning, the time series prediction can be seen as a supervised learning problem, which allows us to express the above problem in a more general form. Given the sets $\mathcal{Y} := \mathbb{R}^{N\times L}$ and $\mathcal{X} := \mathbb{R}^{M\times L}$, sampled from an unknown distribution p, denoted as $\mathcal{D} \in (\mathcal{X} \times \mathcal{Y})$, and a loss function $\mathcal{L} := \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, the goal is to find a function $\mathcal{F} := \mathcal{X} \to \mathcal{Y}$ that minimizes the expected loss

$$\min \mathbb{E}_{((\boldsymbol{x},\boldsymbol{y}),\boldsymbol{y}')\sim p}\big(\mathcal{L}(\boldsymbol{y}', \mathcal{F}(\boldsymbol{x}, \boldsymbol{y}))\big), \tag{1}$$

where $\boldsymbol{x} \in \mathcal{X}$ represents the known sequence, $\boldsymbol{y}, \boldsymbol{y}' \in \mathbf{Y}$ are the predicted and target sequences, respectively, and the function $\mathcal{F}$ can be parameterized as a neural network that takes $\boldsymbol{x}$ as input and produces $\boldsymbol{y}$ as output. The goal is to minimize the expected loss given by the loss function $\mathcal{L}$, considering pairs of predicted and target sequences sampled from the distribution p.

Commonly used neural network architectures for time series modeling include recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and the Transformer [22]. Among them, the Transformer has gained increasing attention due to its ability to capture long-term dependencies and interactions. Compared to RNNs and LSTMs, the Transformer can parallelize the processing of input sequences, providing computational advantages for large-scale data processing. Although these neural network models applied to sequence prediction may differ, they generally share similar architectural forms, as shown in Figure 2.

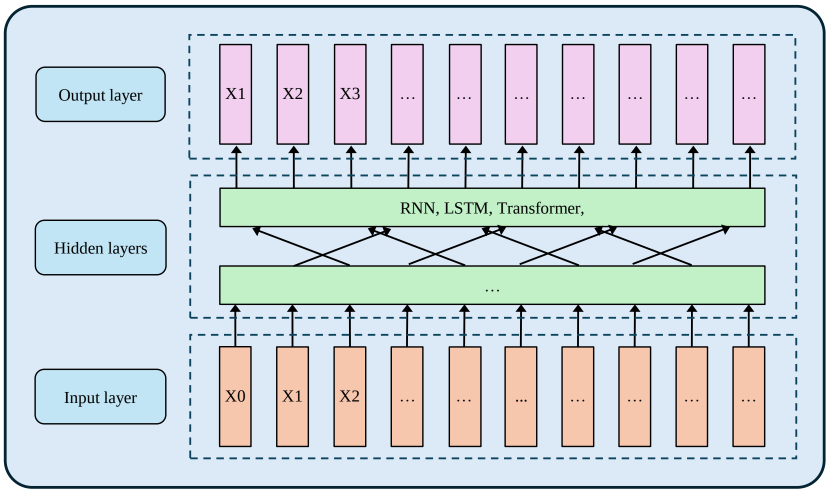


Figure 2. The general architecture of the sequence prediction model.

While these models have the advantages of simplicity and the ability to handle long-term or short-term dependencies, they are typically applied to uniformly sampled time series data. When applied to underwater vehicle motion modeling, it is often required that each state in the sequence has an equal time interval. Although the limitation of non-uniformly sampled data can be addressed by resampling or interpolation, these methods may compromise the original temporal information in the data. To avoid the loss of valuable information, another method is to include time stamps of the time series data as input to the neural network. However, compared to modeling methods based on differential equations, these methods still remain as discrete data modeling and may struggle to effectively model continuous dynamics.

2.2. Neural Ordinary Differential Equations

Neural ODE is a modeling method that combines ordinary differential equations with residual networks (ResNet) [8] to learn the dynamics of a system from data without having to explicitly define the differential equation. Unlike methods such as RNNs, neural ODEs require only the initial state of the system as input, eliminating the need for a sequence of data as input. This feature allows neural ODEs to naturally model time series data with non-uniform sampling and predict the dynamics of a system over continuous time.

As Figure 3 shows, the transformation of hidden states within a neural network can be expressed as

$$h(t+1) = f(h(t), \theta(t)), \tag{2}$$

where f represents a network layer, $h(t)$ and $\theta(t)$ denote the state output and weight matrix of layer t, and $h(t+1)$ represents the state output of layer $t+1$.

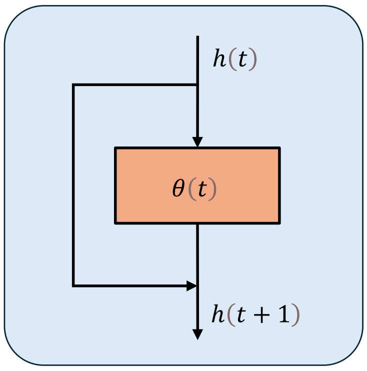


Figure 3. The neural network with residual connection module.

For residual networks, the expression is

$$h(t+1) = h(t) + f(h(t), \theta(t)), \tag{3}$$

where $h(t)$ can skip the network layer at $t+1$ and be added to the output.

To make the connection between the ODE and the residual network, consider a simple ordinary differential equation

$$y'(t) = f(y, t). \tag{4}$$

Given an initial value $y(0) = y_0$ and a time step of 1, the iterative solution process can be expressed as

$$\begin{aligned} y(0) &= y_0 \\ y(1) &= y(0) + y'(0) \\ y(2) &= y(1) + y'(1) \\ &\vdots \\ y(t) &= y(t-1) + y'(t-1) \end{aligned} \tag{5}$$

This process is also known as Euler's method for solving differential equations. If we consider the hidden layer index t in (4) as the time step t in (5), the forward propagation process of a ResNet has the same form as the iterative solution process of a differential equation.

However, there are two differences: (a) when solving an ODE, the time step can take any continuous value, while for ResNet, the time step is discrete, i.e., the number of network layers; (b) different residual blocks of ResNet have different f functions, while in an ODE, there is essentially a single f function defined by (5). Based on this, all residual blocks can be assigned the same parameters. In addition, since the time step is fixed, it can be chosen to be sufficiently small, allowing the network to become deep. As a result, a neural network based on an ODE with continuous depth can be formulated.

2.3. *Hamiltonian Neural Networks*

A neural ODE provides a general method for learning continuous-time dynamics from non-uniformly sampled data without imposing constraints on the underlying dynamical system. This provides versatility but also makes it difficult to accurately model specific physical processes. From a learning theory perspective, this is because a neural ODE has fewer inductive biases and lacks certain necessary assumptions about the target function to be learned.

Hamiltonian mechanics describes the evolution of the system state over time in phase space, where the system state is represented by generalized coordinates

$\boldsymbol{q} = (q_1, q_2, \ldots, q_n) \in \mathbb{R}^n$ and generalized moments $\boldsymbol{p} = (p_1, p_2, \ldots, p_n) \in \mathbb{R}^n$. The Hamiltonian function $H : \mathbb{R}^{2n} \to \mathbb{R}$ maps the state of the system to a scalar representing the total energy of the system. In the context of classical mechanics, the Hamiltonian H represents the sum of the kinetic and potential energies of the system. The Hamiltonian function specifies a vector field in phase space that describes all possible dynamic behaviors of the system. Each state of the system corresponds to a unique trajectory in phase space.

The Hamiltonian function can be expressed as

$$H(\boldsymbol{q},\ \boldsymbol{p}) = T(\boldsymbol{p}) + V(\boldsymbol{q}), \tag{6}$$

where $T(\boldsymbol{p})$ is the kinetic energy and $V(\boldsymbol{q})$ is the potential energy. The evolution of the state of the system over time is described by the Hamiltonian equations

$$\frac{d\boldsymbol{q}}{dt} = \frac{\partial H}{\partial \boldsymbol{p}},\ \frac{d\boldsymbol{p}}{dt} = -\frac{\partial H}{\partial \boldsymbol{q}}. \tag{7}$$

The direction of the vector field defined by (7) is typically called the symplectic gradient of the Hamiltonian system.

It is easy to show that

$$\frac{dH}{dt} = \frac{\partial H}{\partial \boldsymbol{q}}\frac{d\boldsymbol{q}}{dt} + \frac{\partial H}{\partial \boldsymbol{p}}\frac{d\boldsymbol{p}}{dt} = 0. \tag{8}$$

This implies that the total energy of the system is conserved when moving along the direction of the symplectic gradient [23].

In recent years, deep neural networks have been used to learn the Hamiltonian mechanics [24,25]. The Hamiltonian neural network is proposed to model the Hamiltonian function using deep neural networks and satisfy the dynamics of (7), ensuring that the system obeys the law of conservation of energy.

As Figure 4 shows, during forward computation, the HNN takes the generalized coordinates and momenta of the system as input and outputs the total energy of the system H_θ, where θ represents the learnable network parameters. In the backpropagation process, the derivatives of the output with respect to the inputs are computed using automatic differentiation. The loss function of the HNN can be expressed as

$$\mathcal{L}_{HNN} = \left\| \frac{\partial H_\theta}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{q}}{\partial t} \right\|_2 + \left\| \frac{\partial H_\theta}{\partial \boldsymbol{q}} + \frac{\partial \boldsymbol{p}}{\partial t} \right\|_2. \tag{9}$$

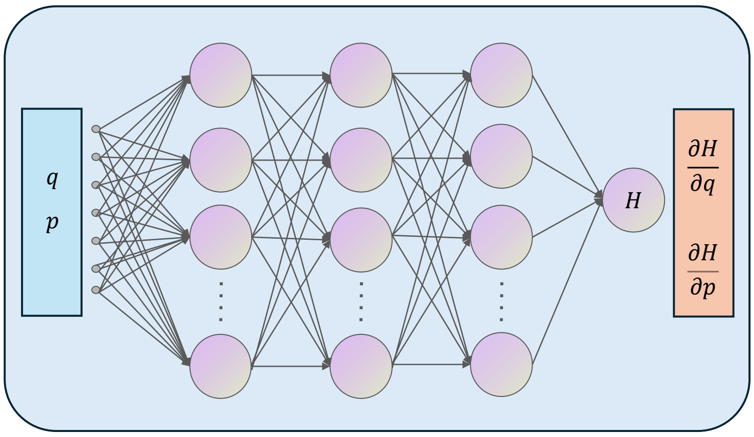
Let

$$S_H(\boldsymbol{q}, \boldsymbol{p}) = \left(\frac{\partial H}{\partial \boldsymbol{p}}, -\frac{\partial H}{\partial \boldsymbol{q}} \right) \tag{10}$$

represent the dynamics of the system's evolution over time. If the system state $(\boldsymbol{q}_0, \boldsymbol{p}_0)$ at time t_0 is known, the neural ODE can be used to obtain the system state at time t_1,

$$(\boldsymbol{q}_1, \boldsymbol{p}_1) = (\boldsymbol{q}_0, \boldsymbol{p}_0) + \int_{t_0}^{t_1} S_H(\boldsymbol{q}, \boldsymbol{p}) dt \tag{11}$$

Compared to the conventional neural ODE, the HNN offers faster training speeds, better generalization performance, and the ability to learn the dynamics of conservative systems more effectively.



q
p
H
$\frac{\partial H}{\partial q}$
$\frac{\partial H}{\partial p}$

Figure 4. The architecture of Hamiltonian neural networks.

3. Motion Modeling Based on Hamiltonian Neural Networks

3.1. Hamiltonian-Based Motion Equation

The kinematic model of an underwater vehicle can be expressed as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{12}$$

where $\boldsymbol{\eta} = (\boldsymbol{p}^n, \boldsymbol{\Theta})^T = (x^n, y^n, z^n, \phi, \theta, \psi)^T$ represents the position and orientation vector in the Earth-fixed coordinate system {n}. Here, the Euler angles ϕ, θ, and ψ correspond to the vehicle's roll, pitch, and yaw, respectively. The vector $\boldsymbol{\nu} = (\boldsymbol{v}, \boldsymbol{\omega})^T = (u, v, w, p, q, r)^T$ represents the velocity vector in the body-fixed coordinate system {b}.

The coordinate transformation matrix $\boldsymbol{J}(\boldsymbol{\eta})$ can be written as

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{pmatrix} \boldsymbol{J}_1 & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{J}_2 \end{pmatrix}, \tag{13}$$

where

$$\boldsymbol{J}_1 = \begin{pmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{pmatrix}, \tag{14}$$

and

$$\boldsymbol{J}_2 = \begin{pmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi / c\theta & c\phi / c\theta \end{pmatrix}. \tag{15}$$

$\boldsymbol{J}_1$ denotes the rotation matrix, which can often be written as $\boldsymbol{R}$, and $c(\cdot)$, $s(\cdot)$, and $t(\cdot)$, respectively, denote $\cos(\cdot)$, $\sin(\cdot)$, and $\tan(\cdot)$.

The dynamic model of an underwater vehicle can be written as

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau}, \tag{16}$$

where $\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A$ represents the inertia matrix, $\boldsymbol{M}_{RB}$ and $\boldsymbol{M}_A$ denote the rigid body mass and hydrodynamic added mass, respectively. $\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu})$ defines the Coriolis and centripetal matrix. $\boldsymbol{D}(\boldsymbol{\nu})$ is the hydrodynamic damping matrix. $\boldsymbol{g}(\boldsymbol{\eta})$ represents the combined gravitational and buoyancy forces, while $\boldsymbol{\tau}$ represents the control force vector.

In the case of small and low-speed underwater vehicles, several hydrodynamic effects can be considered negligible, leading to significant hydrodynamic model simplification [26–28]. According to [29], in an ideal fluid, $\boldsymbol{M}_A$ can be expressed as a semi-positive definite matrix, and this characteristic holds true even in the case of real fluids. In the case of low-speed underwater vehicles, it is often possible to ignore the off-diagonal elements found in the $\boldsymbol{M}_A$ matrix. The term $\boldsymbol{C}_A(\boldsymbol{\nu})$ is always represented as an anti-symmetric matrix, where $\boldsymbol{C}_A(\boldsymbol{\nu}) = -\boldsymbol{C}_A(\boldsymbol{\nu})^T$. The damping terms in high-speed underwater vehicles are highly nonlinear and coupled. Contributions from non-diagonal elements in the damping matrix are also typically negligible for low-speed vehicles.

Due to the influence of hydrodynamics and external control, the underwater vehicle is not a conservative system. Its motion can be described using the Port-Hamiltonian (PH) framework [30]. The PH equation can be written as

$$\begin{pmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{pmatrix} = (\mathbf{J}(\boldsymbol{q},\boldsymbol{p}) - \mathbf{R}(\boldsymbol{q},\boldsymbol{p})) \begin{pmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{pmatrix} + \mathbf{G}(\boldsymbol{q},\boldsymbol{p})\boldsymbol{\tau}_{ctrl}, \tag{17}$$

where $\mathbf{J}(\boldsymbol{q},\boldsymbol{p})$ is an anti-symmetric matrix describing the conservational aspects of the system. $\mathbf{R}(\boldsymbol{q},\boldsymbol{p})$ is a positive definite matrix representing the dissipative effects. $\boldsymbol{\tau}_{ctrl}$ is the control input, and $\mathbf{G}(\boldsymbol{q},\boldsymbol{p})\boldsymbol{\tau}_{ctrl}$ describes external energy inputs. When there are no dissipative terms or external inputs, (17) can degenerate into the standard Hamiltonian equation.

The dynamic model of the underwater vehicle [31], the derivation of which is detailed in Appendix A, can be written as

$$\begin{pmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{pmatrix} = \left(\begin{pmatrix} 0 & \boldsymbol{J} \\ -\boldsymbol{J}^T & -\boldsymbol{C} \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{D} \end{pmatrix} \right) \begin{pmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{pmatrix} + \begin{pmatrix} 0 \\ \boldsymbol{I} \end{pmatrix} \boldsymbol{\tau}, \tag{18}$$

where $\boldsymbol{q} = \boldsymbol{\eta}$, $\boldsymbol{p} = \boldsymbol{M}\boldsymbol{\nu}$, and the Hamiltonian function can be expressed as

$$H = \frac{1}{2}\boldsymbol{p}^T\boldsymbol{M}^{-1}\boldsymbol{p} + V(\boldsymbol{q}). \tag{19}$$

3.2. Representation of Rotational Motion

The orientation of underwater vehicles is typically represented by Euler angles. To unify the generalized coordinates in real space, the angle coordinates $\boldsymbol{q}$ can be represented as a two-dimensional embedding $(\cos\boldsymbol{q}, \sin\boldsymbol{q})$ [23,32]. However, when dealing with the rotation of underwater vehicles, the exponential map provides a more effective solution. The exponential map is a method that maps angular velocity vectors to their corresponding rotation matrices and can be used to update the orientation of the underwater vehicle.

The advantages of using the exponential map to represent rotations are as follows: (a) ensuring orthogonality and normalization of the rotation matrices, preventing the accumulation of numerical errors; (b) maintaining energy conservation within the system, avoiding energy drift; (c) avoiding coordinate singularities and local minimization problems, improving numerical stability and accuracy; (d) conveniently representing small-angle rotations of rigid bodies, improving computational efficiency.

By using the exponential map, changes in the orientation of underwater vehicles can be more accurately and stably modeled and updated. This method has significant value in simulating the motion of underwater vehicles, as well as in tasks such as control and path planning, and offers important applications in various scenarios.

The rotation matrix $\boldsymbol{R}$ in (14) can be represented as $\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)$, where $\boldsymbol{R}$ belongs to the special orthogonal group SO(3), which is defined as

$$\mathrm{SO}(3) = \left\{ \boldsymbol{R} \in \mathbb{R}^{3\times 3} \middle| \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I},\ \det \boldsymbol{R} = 1 \right\}. \tag{20}$$

The derivative of the rotation matrix $\boldsymbol{R}(t)$ with respect to time can be expressed as

$$\dot{\boldsymbol{R}}(t) = \boldsymbol{R}(t)\boldsymbol{\omega}(t)^{\wedge}, \tag{21}$$

where $(\cdot)^{\wedge}$ represents the skew-symmetric matrix, and $\boldsymbol{\omega}^{\wedge}$ corresponds to the Lie algebra $\mathfrak{so}(3)$ associated with SO(3), which is given by

$$\boldsymbol{\omega}^{\wedge} = \begin{pmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{pmatrix}. \tag{22}$$

Solving the differential equation in (21), we can express the orientation at time t based on its initial state $R(t_0)$ and the angular velocity history. Assuming $t_0 = 0$ and a constant angular velocity ω_0 over a small time step, the solution is given by the matrix exponential:

$$R(t) = R(0)\exp(\omega_0^{\wedge} t) \tag{23}$$

It is important to note that the solution depends on the initial orientation $R(0)$. In many simulation and control scenarios, including those in this work, the vehicle is assumed to start from a reference orientation, for which $R(0) = I$, where I is the 3×3 identity matrix.

If the translational motion of an underwater vehicle is also considered, a transformation matrix $\boldsymbol{T}$ can be defined. $\boldsymbol{T}$ belongs to the special Euclidean group SE(3) and is defined as

$$\mathrm{SE}(3) = \left\{ \boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{p}^n \\ \boldsymbol{0}^T & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4} \middle| \boldsymbol{R} \in \mathrm{SO}(3),\ \boldsymbol{p}^n \in \mathbb{R}^3 \right\}. \tag{24}$$

The derivative of transformation matrix $\boldsymbol{T}$ with respect to time can be expressed as

$$\dot{\boldsymbol{T}} = \boldsymbol{T}\boldsymbol{\nu}^{\wedge} = \boldsymbol{T}\begin{pmatrix} \boldsymbol{\omega}^{\wedge} & \boldsymbol{v} \\ \boldsymbol{0}^T & 0 \end{pmatrix}, \tag{25}$$

where $(\cdot)^{\wedge}$ extends the definition of the skew-symmetric matrix to transform a six-dimensional vector into a four-dimensional matrix, and $\boldsymbol{\nu}^{\wedge}$ corresponds to the Lie algebra $\mathfrak{se}(3)$ associated with SE(3).

3.3. *Hamiltonian-Based Motion Model for Underwater Vehicles*

After representing the attitude of the underwater vehicle using the rotation matrix instead, the generalized coordinates can be expressed as

$$\boldsymbol{q} = \left(\boldsymbol{p}^{nT}, \boldsymbol{r}_1^T, \boldsymbol{r}_2^T, \boldsymbol{r}_3^T\right)^T \in \mathbb{R}^{12}. \tag{26}$$

If $\boldsymbol{\nu} = \left(\boldsymbol{v}^T, \boldsymbol{\omega}^T\right)^T$ continues to represent the generalized velocity, then the dimensions of the generalized coordinates $\boldsymbol{q}$ and generalized momentum $\boldsymbol{p}$ are different, but they still satisfy the constraint condition of (25). The derivative of the generalized coordinates $\boldsymbol{q}$ in this case can be expressed as $\dot{\boldsymbol{q}} = \boldsymbol{q}^{\times}\boldsymbol{\nu}$ [33], where

$$\boldsymbol{q}^{\times} = \begin{pmatrix} \boldsymbol{R}^T & 0 & 0 & 0 \\ 0 & \boldsymbol{r}_1^{\wedge T} & \boldsymbol{r}_2^{\wedge T} & \boldsymbol{r}_3^{\wedge T} \end{pmatrix}^T \in \mathbb{R}^{12\times 6}. \tag{27}$$

The Lagrangian function on the SE(3) manifold can be written as

$$L(\boldsymbol{q}, \boldsymbol{\nu}) = \frac{1}{2}\boldsymbol{\nu}^T \boldsymbol{M}(\boldsymbol{q})\boldsymbol{\nu} - \boldsymbol{V}(\boldsymbol{q}). \tag{28}$$

The generalized momentum in Hamiltonian form can be described as the partial derivative of the Lagrangian function with respect to the generalized velocity, as follows:

$$\boldsymbol{p} = \frac{\partial L}{\partial \boldsymbol{v}} = \boldsymbol{M}(\boldsymbol{q})\boldsymbol{v} \in \mathbb{R}^6. \tag{29}$$

Let $\boldsymbol{p} = \left(\boldsymbol{p}_v^T, \boldsymbol{p}_\omega^T\right)^T$, and by comparing (17) and (18), we have

$$\begin{aligned} \mathbf{J}(\boldsymbol{q},\boldsymbol{p}) &= \begin{pmatrix} \mathbf{0} & \boldsymbol{q}^{\times} \\ -\boldsymbol{q}^{\times T} & \boldsymbol{p}^{\times} \end{pmatrix} \\ \mathbf{R}(\boldsymbol{q},\boldsymbol{p}) &= \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{D}(\boldsymbol{q}) \end{pmatrix} \quad , \\ \mathbf{G}(\boldsymbol{q},\boldsymbol{p}) &= \begin{pmatrix} \mathbf{0} \\ \mathbf{g}(\boldsymbol{q}) \end{pmatrix} \end{aligned} \tag{30}$$

where

$$\begin{aligned} \boldsymbol{p}^{\times} &= \begin{pmatrix} \mathbf{0} & \boldsymbol{p}_v^{\wedge} \\ \boldsymbol{p}_v^{\wedge} & \boldsymbol{p}_\omega^{\wedge} \end{pmatrix} \in \mathbb{R}^{6\times 6} \\ \boldsymbol{D}(\boldsymbol{q}) &= \begin{pmatrix} \boldsymbol{D}_{11} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \boldsymbol{D}_{22} \end{pmatrix} \in \mathbb{R}^{6\times 6}. \\ \mathbf{g}(\boldsymbol{q}) &= \begin{pmatrix} \mathbf{g}_{\mathbf{v}} \\ \mathbf{g}_{!} \end{pmatrix} \in \mathbb{R}^6 \end{aligned} \tag{31}$$

Substituting (30) into (18), we obtain

$$\begin{aligned} \dot{\boldsymbol{p}^n} &= \boldsymbol{R}\frac{\partial H}{\partial \boldsymbol{p}_v} \\ \dot{\boldsymbol{r}_i} &= \boldsymbol{r}_i \times \frac{\partial H}{\partial \boldsymbol{p}_\omega}, \quad i = 1,2,3 \\ \dot{\boldsymbol{p}_v} &= \boldsymbol{p}_v \times \frac{\partial H}{\partial \boldsymbol{p}_\omega} - \boldsymbol{R}^T\frac{\partial H}{\partial \boldsymbol{p}^n} - \boldsymbol{D}_{11}\frac{\partial H}{\partial \boldsymbol{p}_v} + \mathbf{g}_v(\boldsymbol{q})\boldsymbol{u} \\ \dot{\boldsymbol{p}_\omega} &= \boldsymbol{p}_v \times \frac{\partial H}{\partial \boldsymbol{p}_v} + \boldsymbol{p}_\omega \times \frac{\partial H}{\partial \boldsymbol{p}_\omega} - \sum \boldsymbol{r}_i \times \frac{\partial H}{\partial \boldsymbol{r}_i} \\ &\quad - \boldsymbol{D}_{22}\frac{\partial H}{\partial \boldsymbol{p}_\omega} + \mathbf{g}_\omega(\boldsymbol{q})\boldsymbol{u} \end{aligned} \tag{32}$$

The derivative of the generalized velocity with respect to time can be obtained from the Lagrangian function

$$\dot{\boldsymbol{v}} = \left(\frac{d}{dt}\boldsymbol{M}^{-1}(\boldsymbol{q})\right)\boldsymbol{p} + \boldsymbol{M}^{-1}(\boldsymbol{q})\dot{\boldsymbol{p}}. \tag{33}$$

Consequently, the complete Hamiltonian-based dynamic model of the underwater vehicle can be represented using (32) and (33).

3.4. Training Strategy

3.4.1. Definition of the Loss Function

Let $\mathcal{D}$ represent a dataset containing M motion trajectories,

$$\mathcal{D} = \left\{t_{0:N}^{(i)}, \boldsymbol{x}_{0:N}^{(i)}, \boldsymbol{u}^{(i)}\right\}_{i=1}^{M}, \tag{34}$$

where each trajectory includes a sequence of motion states under the influence of control inputs $\boldsymbol{u}^{(i)}$ at time steps $t_{0:N}^{(i)}$.

The system state at time $t_n^{(i)}$ can be written as

$$\begin{aligned}\boldsymbol{x}^{(i)}(t_n) = \boldsymbol{x}_n^{(i)} &= \left(\boldsymbol{q}_n^{(i)^T}, \boldsymbol{\nu}_n^{(i)^T}\right)^T \\ &= \left(\boldsymbol{p}_n^{n(i)^T}, \boldsymbol{r}_{1n}^{(i)^T}, \boldsymbol{r}_{2n}^{(i)^T}, \boldsymbol{r}_{3n}^{(i)^T}, \boldsymbol{v}_n^{(i)^T}, \boldsymbol{\omega}_n^{(i)^T}\right)^T \in \mathbb{R}^{18}.\end{aligned} \tag{35}$$

Let $\widetilde{\boldsymbol{x}}_n^{(i)}$ denote the predicted motion state at time $t_n^{(i)}$ given an initial state $\boldsymbol{x}_0^{(i)}$ and control input $\boldsymbol{u}^{(i)}$. The loss function for the model can be defined as

$$\mathcal{L} = \mathcal{L}(\boldsymbol{p}^n) + \mathcal{L}(\boldsymbol{R}) + \mathcal{L}(\boldsymbol{\nu}). \tag{36}$$

The first part concerns the prediction error of the underwater vehicle's position $\boldsymbol{p}^n$,

$$\mathcal{L}(\boldsymbol{p}^n) = \sum \left\| \widetilde{\boldsymbol{p}}_n^{n(i)} - \boldsymbol{p}_n^{n(i)} \right\|_2 \tag{37}$$

The second part addresses the prediction error of the rotation matrix $\boldsymbol{R}$,

$$\mathcal{L}(\boldsymbol{R}) = \sum \left\| (\log \widetilde{\boldsymbol{R}}_n^{(i)} \boldsymbol{R}_n^{(i)^T})^\vee \right\|_2, \tag{38}$$

where $\log(\cdot)$ denotes the logarithmic mapping from SO(3) to $\mathfrak{so}(3)$, and $(\cdot)^\vee$ represents the inverse operation of $(\cdot)^\wedge$ ([34]).

The third part focuses on the prediction error of the underwater vehicle's velocity $\boldsymbol{\nu}$,

$$\mathcal{L}(\boldsymbol{\nu}) = \sum \left\| \widetilde{\boldsymbol{\nu}}_n^{(i)} - \boldsymbol{\nu}_n^{(i)} \right\|_2. \tag{39}$$

3.4.2. Selection of Integration Scheme

When simulating Hamiltonian systems over long horizons, the choice of numerical integrator is critical. Non-symplectic algorithms, such as the standard Euler method, often introduce numerical dissipation that leads to an artificial drift in the system's total energy, accumulating significant errors over time. In contrast, symplectic integrators are specifically designed to preserve the geometric structure of Hamiltonian dynamics, which makes them far more stable for long-term predictions [35]. For this reason, we select the widely used, second-order symplectic algorithm, the Leapfrog integrator, for solving the learned ODEs.

A key parameter for any integrator is the time step, δt. In our implementation, a deliberate choice is made to match the integration time step with the sampling period of the trajectory data. While it is common practice to use an integration step significantly smaller than the data sampling period, our approach is justified by the physical characteristics of the system under study. The AUV dynamics, particularly in the low-speed maneuvers considered, are sufficiently smooth and non-stiff, allowing for stable and accurate integration without requiring a finer time step. This methodology provides a significant advantage in computational efficiency. The specific parameter values are detailed in Section 4.1, and the successful long-term prediction results in our experiments serve as strong empirical validation for this approach.

The Leapfrog integration scheme is expressed as follows:

$$\begin{aligned} \boldsymbol{p}_{n+\frac{1}{2}} &= \boldsymbol{p}_n + \frac{1}{2}\delta t \frac{d\boldsymbol{p}_n}{dt} \\ \boldsymbol{q}_{n+1} &= \boldsymbol{q}_n + \delta t \frac{d\boldsymbol{q}_{n+\frac{1}{2}}}{dt} \\ \boldsymbol{p}_{n+1} &= \boldsymbol{p}_{n+\frac{1}{2}} + \frac{1}{2}\delta t \frac{d\boldsymbol{p}_{n+1}}{dt} \end{aligned} \tag{40}$$

where the subscript n corresponds to the iteration step. This algorithm is more precise than the Euler method for Hamiltonian systems and effectively alleviates the problem of unstable error growth during multi-step integration.

4. Experiment and Results

4.1. Experimental Setup

The proposed HNN-based motion modeling method is validated using the REMUS 100 underwater vehicle, which is depicted in Table 1 [29]. The REMUS 100 achieves axial propulsion through a stern thruster and maneuverability via symmetrically arranged sets of rudders and elevators.

Table 1. Parameters of REMUS 100 AUV.

No.	Name	Value	Unit
1	Length	1.6	m
2	Diameter	0.19	m
3	Mass	31.9	kg
4	Maximum speed	2.5	m/s
5	Center of gravity position	(0, 0, 0.02)	m
6	Center of buoyancy position	(0, 0, 0)	m
7	Propeller diameter	0.14	m
8	Propeller maximum speed	1525	rpm
9	Rudder longitudinal position	−0.85	m
10	Rudder maximum angle	30	deg
11	Elevator longitudinal position	−0.85	m
12	Elevator maximum angle	30	deg

To model the Hamiltonian equations for the underwater vehicle, we employ four independent neural networks to learn the system's constituent matrices: the generalized mass matrix $M(\boldsymbol{q})$, the potential energy $V(\boldsymbol{q})$, the damping matrix $D(\boldsymbol{q})$, and the control input matrix $\boldsymbol{g}(\boldsymbol{q})$. All networks share a common architecture, consisting of two hidden layers with 128 nodes each and using the Tanh activation function. Furthermore, they are all conditioned on the same input: the 12-dimensional generalized coordinate vector $\boldsymbol{q} = (\boldsymbol{p}^{nT}, \boldsymbol{r}_1^T, \boldsymbol{r}_2^T, \boldsymbol{r}_3^T)^T$, which fully describes the vehicle's position and orientation.

The output structure of each network is tailored to incorporate prior physical knowledge about the vehicle, which is a strategy that significantly reduces the dimensionality of the learning problem and improves training efficiency. For instance, based on the vehicle's known symmetries and its operation in a low-speed regime, the $M(\boldsymbol{q})$ network is designed to output only seven scalar values, corresponding to the six principal diagonal elements and the dominant off-diagonal coupling term (M_{15}). Similarly, the $D(\boldsymbol{q})$ network learns the six diagonal elements of the hydrodynamic damping matrix. The $V(\boldsymbol{q})$ network outputs a single scalar for the system's potential energy, and the $\boldsymbol{g}(\boldsymbol{q})$ network learns an 18-element vector that forms a 6×3 matrix, mapping the three control inputs (propeller revolution, rudder angle, and elevator angle) to the corresponding forces and torques in the 6 degrees of freedom.

It is crucial to emphasize that these output simplifications are a practical application of prior knowledge—not an inherent limitation of our framework. The proposed HNN-based methodology is fully capable of learning complete, dense mass and damping matrices for vehicles with more complex or less understood dynamics provided that sufficient and adequately exciting training data are available.

4.2. Experimental Result

The REMUS 100 motion model provided by [29] was used to generate training data. One hundred motion trajectories of 5 s each were generated under random control inputs, starting from the initial state of $\boldsymbol{q} = \mathbf{0}$ and $\boldsymbol{p} = \mathbf{0}$. The control and motion sampling frequencies were 10 Hz and 50 Hz, respectively. Following this, non-overlapping samples of 0.1 s each were obtained by randomly cropping 3 s segments from each trajectory while keeping the control input unchanged. We obtained a total of 3000 data samples through this process. The data were randomly split into training and testing sets at an 8:2 ratio.

All models were implemented using the PyTorch framework. The training and evaluation were conducted on a workstation equipped with a single NVIDIA RTX 3090 GPU. For training, we used the Adam optimizer with default parameters and a batch size of 128. The learning rate was set to 5×10^{-4} and was kept constant throughout the 500 training epochs. To ensure the statistical significance and robustness of our results, all experiments were repeated five times using different random seeds for data shuffling and network initialization. Figure 5 shows the variation of the loss on the dataset during the training.

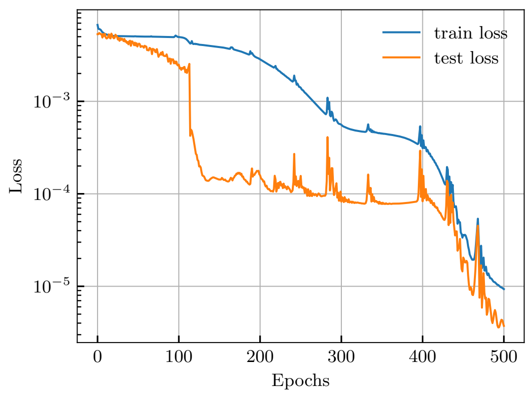


Figure 5. Loss variation curves during training.

The loss on the training set decreases consistently with a more rapid decline in the final 100 epochs. At the same time, the loss on the test set remains consistently lower than that on the training set, although its trend is more complex. Starting from the 100th epoch, the test loss experiences a rapid decrease to the 1×10^{-3} level, which is followed by almost constant maintenance with intense fluctuations over the next 250 epochs. Nonetheless, during the final 100 epochs, the test loss set decreases rapidly again, eventually approaching the 1×10^{-6} level. Although the loss curve is not always smooth, the model's loss reduces to a very low level after 500 epochs of training, indicating successful completion.

To more intuitively evaluate the training results, we generated 100 random motion trajectories of 10 s each using the same method described earlier. Next, we used the trained model to predict changes in the motion state of the vehicle from the same initial state within 10 s. It is important to note that the model has not predicted continuous trajectories beyond 0.1 s during training. As a result, this evaluation provides a better test of the model's generalization ability.

Figure 6 shows the predicted generalized mass matrix by the trained model, where the learned elements remain nearly constant throughout the 10 s prediction. A remarkable result is the clear symmetry discovered by the network, where $M_{22} \approx M_{33}$ and $M_{55} \approx M_{66}$. This is not a trivial consequence of the model's architecture but rather a meaningful demonstration of learning physical consistency.

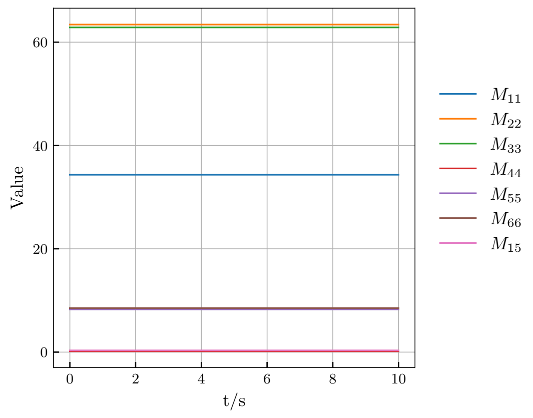


Figure 6. The value of the generalized mass matrix.

Crucially, the neural network architecture imposes no prior structural constraints that would force these matrix elements to be equal; each output is parameterized and learned independently. The observed symmetry is a direct reflection of the physical properties of the REMUS 100 AUV, which, as an axisymmetric vehicle, exhibits an identical hydrodynamic response to motions in the sway and heave directions. The fact that our model autonomously discovered and internalized this fundamental physical principle purely from observing trajectory data serves as strong validation. It shows that the framework is genuinely learning the underlying system physics, distinguishing it from a simple black-box curve-fitter.

Figure 7 shows the variation of two variables associated with the rotation matrix over time, whose expected values are zero. Despite a gradual increase over time, the predicted values provided by the Hamiltonian neural network remain small with their absolute value peaking at around 1×10^{-7}. Therefore, it can be concluded that these values stay within a certain range. Our proposed method based on the Hamiltonian neural network does not rely on complex physical priors and parameter estimation, as opposed to traditional modeling methods based on differential equations. Despite its relatively simple structure, it efficiently captures the dynamic characteristics of underwater vehicles. Furthermore, the high accuracy in predicting rotational motion in the long term is still evident, which will be further demonstrated in subsequent case studies.

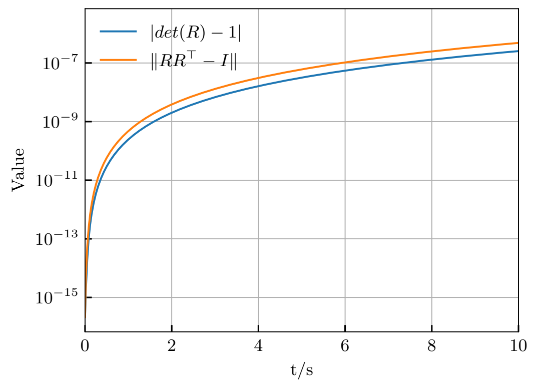


Figure 7. The variation of variables associated with the rotation matrix.

Figure 8 shows the predicted changes in the Hamiltonian over time. The Hamiltonian is not constant in this case; however, its fluctuations are very small. This implies that the model has learned the Hamiltonian's invariance. With increasing prediction time, these fluctuations become larger, although they still remain confined to a very narrow range overall. This observation demonstrates that predicting the long-term motion of underwater vehicles is a challenging task, as errors tend to accumulate over time.

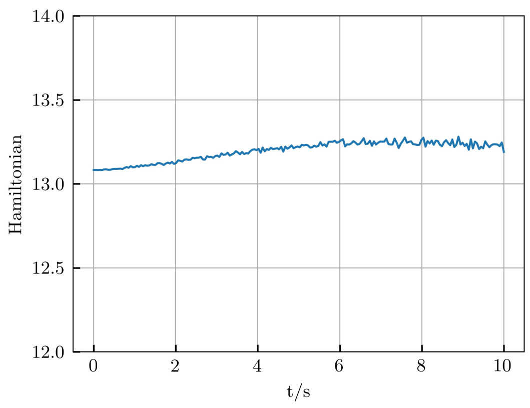


Figure 8. The variation of Hamiltonian over time.

4.3. Case Study

This section presents a comparison between our HNN method and another one based on temporal convolutional networks (TCNs) [36]. The TCN takes as input the historical motion state information of the underwater vehicle over a certain time window. It employs multiple layers of temporal convolutional operations to compress the high-dimensional input, extracting effective low-dimensional features in chronological order. The extracted features are passed through several fully connected layers with nonlinear activation functions to produce the acceleration information for the current moment. The TCN has a simpler structure than our HNN. However, after extensive training with a massive amount of data, it can also demonstrate strong performance. We trained the TCN using the dataset mentioned earlier and implemented it according to the guidelines presented in [36].

4.3.1. Simple Scenarios

Figure 9 shows how the TCN and HNN perform when predicting the straight-line motion of underwater vehicles. In this scenario, the rudder angles for both yaw and pitch are held steady at 0 degrees, while the propeller's rotational speed gradually increases to 1500 rpm. Both neural network-based methods effectively simulate the vehicle's motion with predictions of remarkably similar precision. The accuracy of the TCN slightly surpasses that of the HNN. Notably, due to the coupling effects of motion, the vehicle does not strictly move in a straight line along the inertial X-axis. Instead, it exhibits slight deviations in the Y and Z-axis directions, which stabilize over time.

Figure 10 shows the variation in the motion state of the underwater vehicle after it is steered 10, 20, and 30 degree to the right while maintaining a propeller speed of 1500 rpm. The initial two columns depict the positional changes of the vehicle over time, and the third column demonstrates the variations in the yaw angle. The predictive outcomes of both neural network-based methods are almost identical. This suggests that both the TCN and HNN are skilled in anticipating the vehicle's turning motion, thereby upholding a high degree of predictive accuracy, even in intricate scenarios.

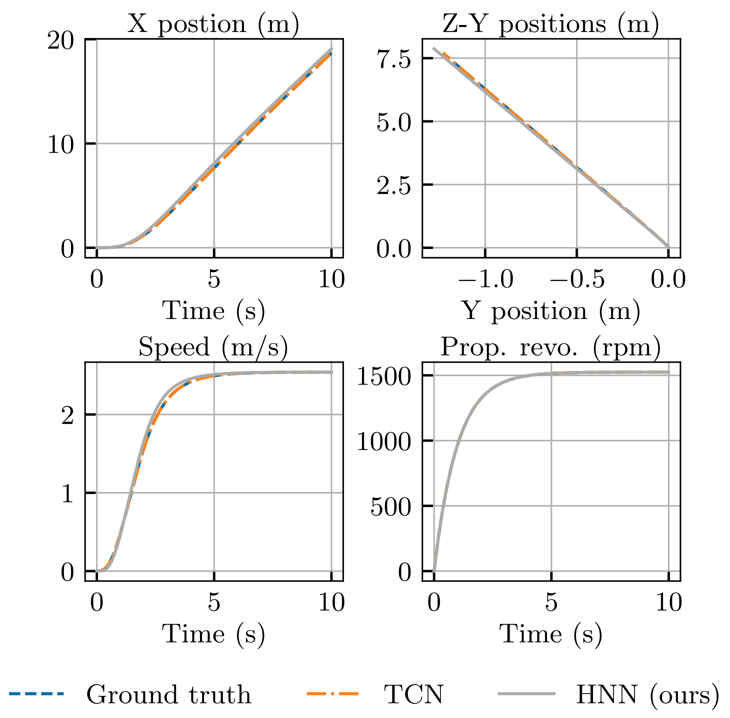


Figure 9. Prediction of the linear motion of REMUS 100 using two models.

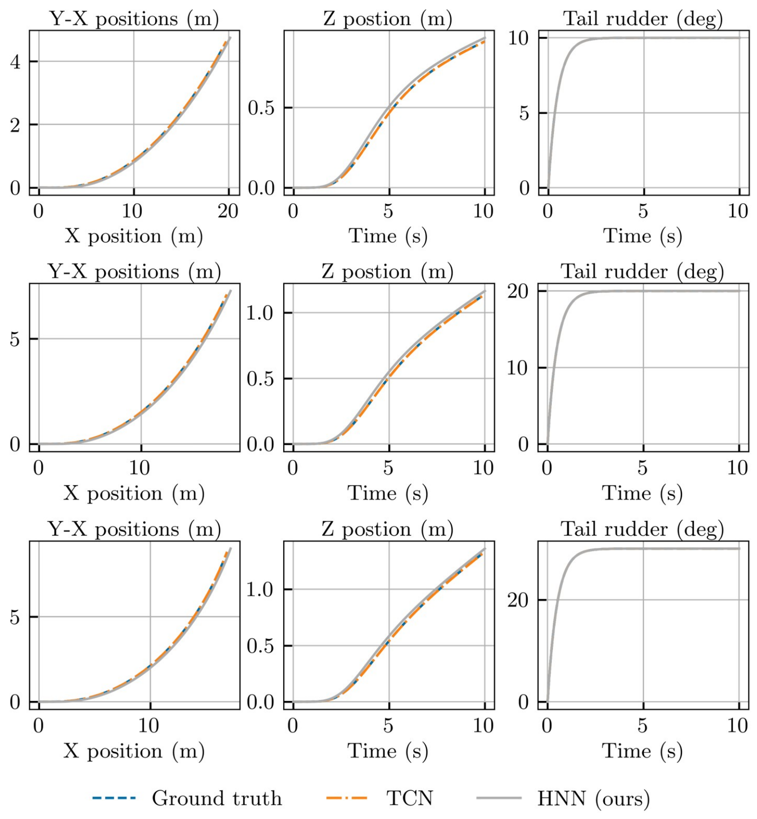


Figure 10. Motion of REMUS 100 was observed after turning the rudder to 10°, 20°, and 30° toward the right.

Figure 11 illustrates the motion of the underwater vehicle as the pitch angle is adjusted by 5, 10, and 15 degrees while maintaining a constant propeller speed of 1500 rpm. The vehicle initiates a descent in this scenario. The first two columns display the temporal evolution of the vehicle's position, and the third column demonstrates the changes in the pitch angle. Both methods adeptly model the descent motion of the underwater vehicle with predictive outcomes closely aligning with benchmark data.

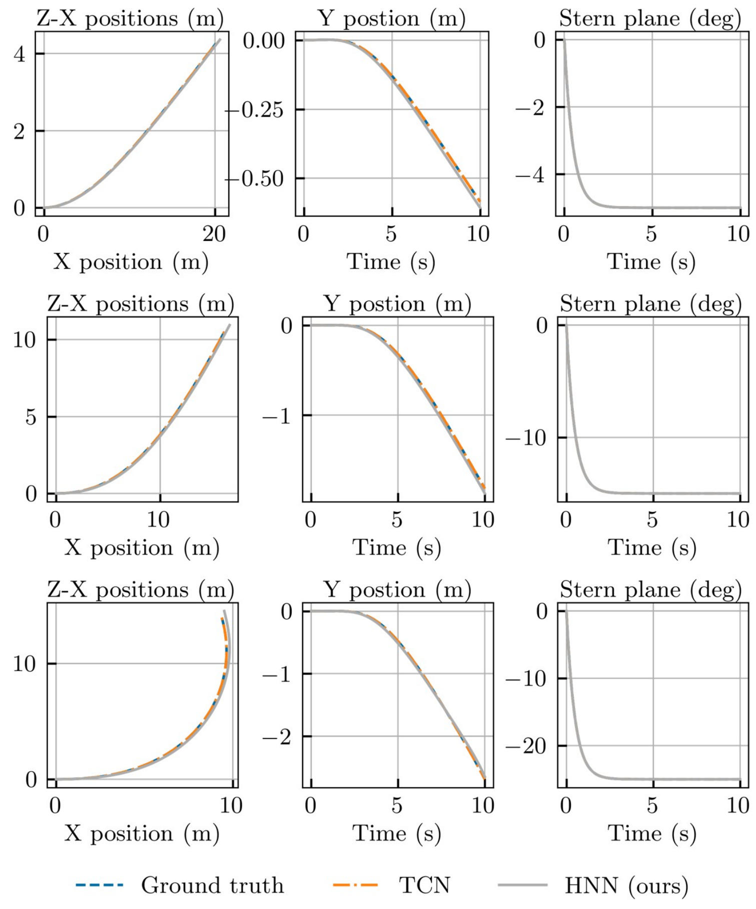


Figure 11. Motion of REMUS 100 was observed after lowering the elevator by 5°, 15°, and 25°.

4.3.2. Complex Scenarios

The above analysis shows that the TCN method performs exceptionally well in simple scenarios with straight-line and turning motions even in the absence of physical priors. However, to demonstrate the benefits of the proposed HNN method, we analyze more complex motion scenarios.

Figure 12 shows the changes in the motion state of the underwater vehicle due to random controls. The first two rows demonstrate the vehicle's orientation in three-dimensional space, and the following two rows show the variations in velocity and angular velocity over time. Figure 13 shows the control inputs corresponding to Figure 12, where dashed lines represent desired control generated randomly, and solid lines represent the applied inputs.

As the complexity of the control inputs increases, the TCN method maintains relatively accurate predictions of the motion of the underwater vehicle for a short time. However, approximately 2 s after the initiation of motion, TCN's predictions of velocity vectors other than the longitudinal velocity start to deviate from the benchmark. This results in a gradual difference between the predicted and the actual motion of the vehicle. It is important to note that the TCN method demonstrated exceptional performance in the simple motion scenarios that came before.

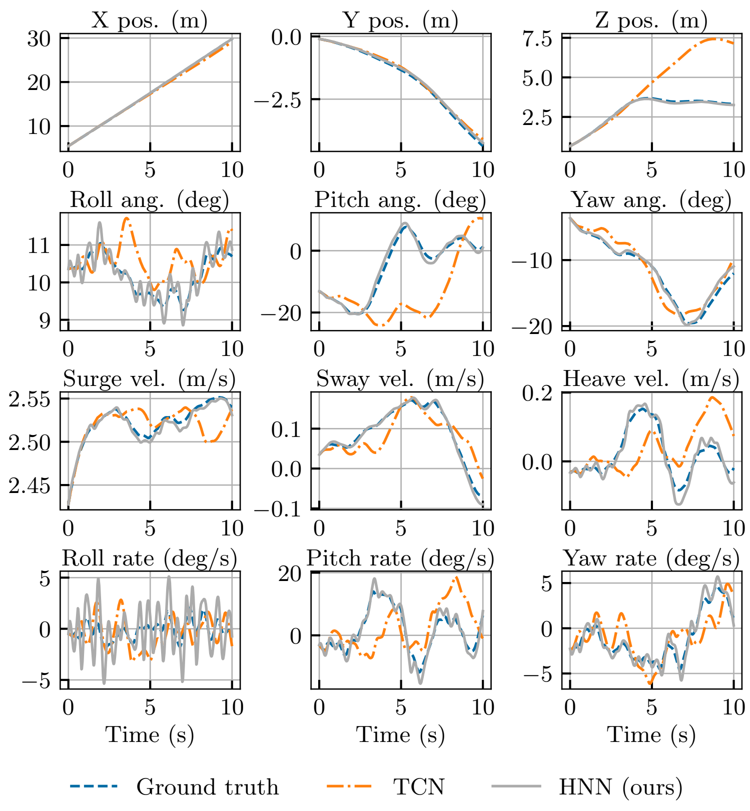


Figure 12. Motion of REMUS 100 influenced by random external control.

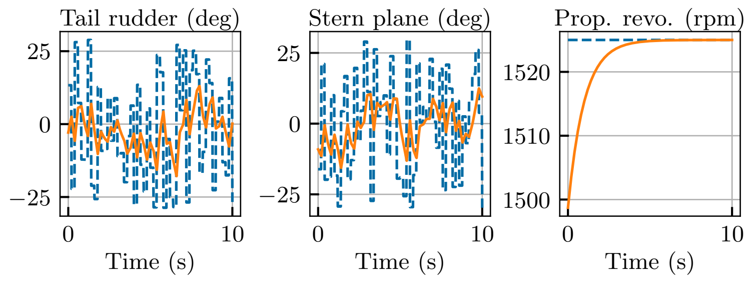


Figure 13. Random external control acting on the REMUS 100, where dashed lines represent control commands and solid lines represent actual control inputs.

Although the HNN method did not demonstrate superior performance to the TCN in previous scenarios, it now outperforms the TCN's performance in the more complex control scenario. In spite of HNN's minor overestimation in predicting the vehicle's velocity, it consistently delivers highly accurate predictions of the vehicle's pose that closely match the actual outcomes. This emphasizes that the HNN-based motion modeling approach, which is based on robust physical principles, better captures the dynamic characteristics of underwater vehicles, going beyond mere data fitting. The HNN method maintains consistent predictive accuracy across a spectrum of control inputs that range from simple to complex.

Figures 14–17 show the motion variation in response to two additional sets of random control inputs. The TCN method maintains high accuracy only during the initial stages of motion, which is followed by a sharp decline in performance. In contrast, our HNN maintains high accuracy consistently and produces predictive outcomes that closely match the true motion. It is worth noting that both the TCN and HNN provide relatively accurate predictions of the longitudinal motion of the underwater vehicle. This is attributed to underwater vehicles such as REMUS 100, which have torpedo-like profiles and predominantly move longitudinally, accounting for a considerable part of all motion components. Through extensive data training, the TCN method is also capable of capturing this leading motion pattern. However, the proposed HNN method, which is built on the physical principles of underwater vehicle motion, consistently performs well in predicting all motion components. Therefore, the HNN outperforms in various scenarios.

While the trajectory plots in Figures 14–17 provide a compelling qualitative illustration of the HNN's superior robustness, a quantitative analysis is necessary to rigorously assess the performance gap between the two models. To this end, we conducted a series of 10 tests using unique, randomly generated complex control inputs, mirroring the conditions in the qualitative examples. In each test, we evaluated the 10 s prediction performance of both the HNN and TCN models against the ground truth. The root mean square error (RMSE) was calculated across the entire trajectory for key state variables.

The aggregated results of this analysis are presented in Table 2. The data reveal a dramatic and unambiguous difference in performance. The HNN model consistently achieves extremely low prediction errors across all metrics, with a mean position error of only 3.3 cm and very low variance, indicating its stable and reliable performance across different chaotic scenarios. In stark contrast, the TCN model's predictions diverge significantly, resulting in a mean position error of over 5.4 m—more than 160 times greater than that of the HNN. Its errors for attitude, linear velocity, and angular velocity are also one to two orders of magnitude higher than those of our model.

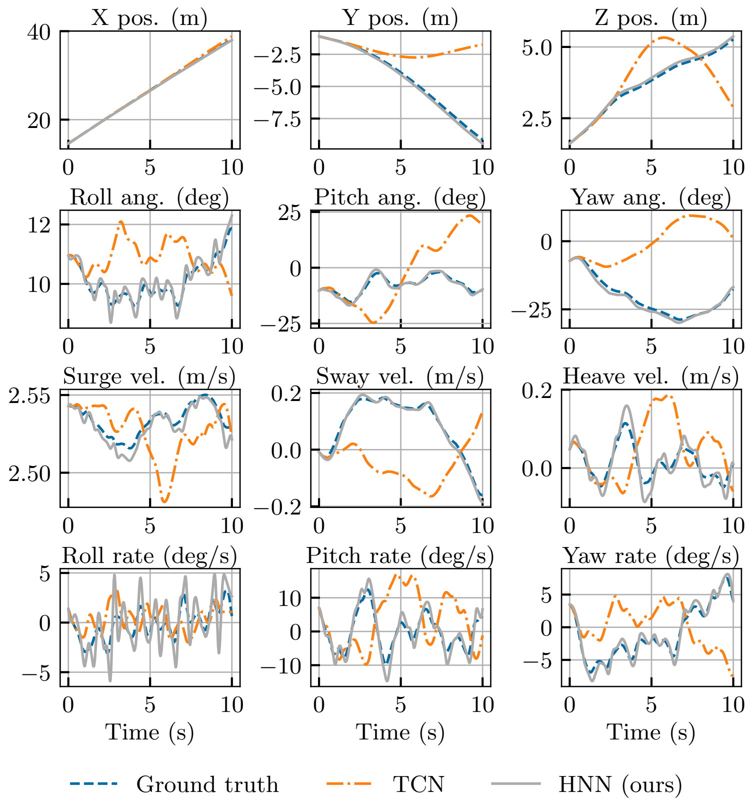


Figure 14. Motion resulting from the 2nd set of random control.

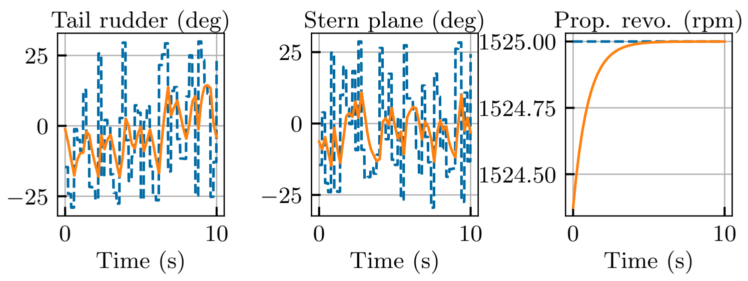


Figure 15. The 2nd set of random control acting on the REMUS 100, where dashed lines represent control commands and solid lines represent actual control inputs.

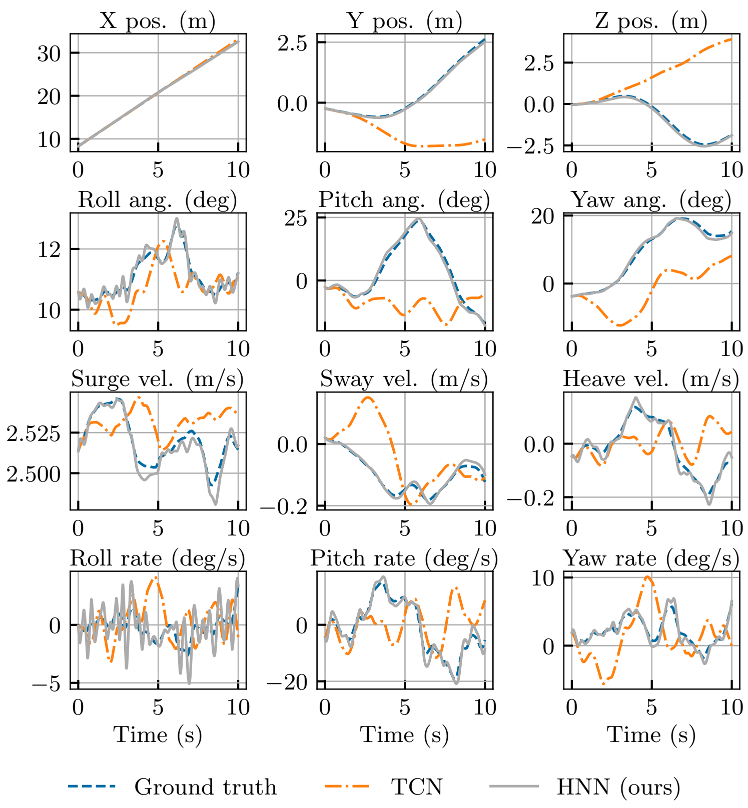


Figure 16. Motion resulting from the 3rd set of random controls.

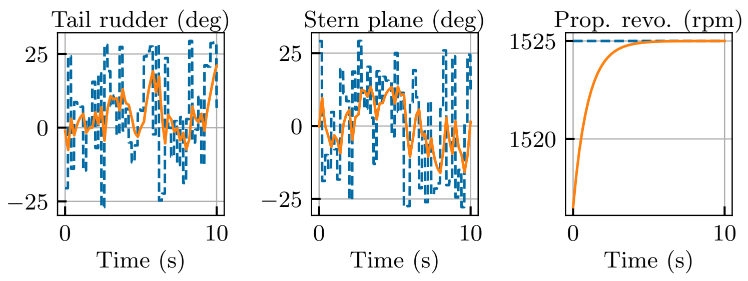


Figure 17. The 3rd set of random control acting on the REMUS 100, where dashed lines represent control commands and solid lines represent actual control inputs.

This quantitative analysis provides conclusive evidence that the physics-informed structure of the HNN is essential for maintaining long-term prediction stability and accuracy when faced with complex, out-of-distribution control inputs. Under these challenging conditions, the purely data-driven TCN model, despite its strong performance in simple scenarios, proves to be unreliable and fails catastrophically.

Table 2. Quantitative comparison of 10 s prediction RMSE for HNN and TCN models, aggregated over 10 different complex random control scenarios. Values are presented as mean ± standard deviation.

Metric	**HNN (Ours) (Mean ± Std Dev)**	**TCN (Mean ± Std Dev)**
Position [m]	0.033 ± 0.012	5.419 ± 0.026
Attitude [deg]	0.490 ± 0.169	5.999 ± 1.842
Linear Velocity [m/s]	0.007 ± 0.002	0.141 ± 0.005
Angular Velocity [deg/s]	0.659 ± 0.128	4.771 ± 0.678

5. Discussion

The experimental results offer a clear narrative about the differing capabilities of purely data-driven and physics-informed models, highlighting the critical distinction between interpolation and physical generalization.

5.1. The Power of Inductive Bias for Generalization

Our findings reveal a fundamental divergence in model capabilities: the TCN, as a high-capacity black-box model, excels at *interpolation*, whereas our HNN demonstrates superior *extrapolation*. The TCN's slightly better performance in simple scenarios is a direct consequence of these test cases mirroring the constant-control structure of the training data. It is an excellent pattern-matcher for on-distribution tasks. However, this is also its critical weakness. As quantitatively demonstrated in Table 2, the TCN's performance collapses catastrophically when faced with complex, out-of-distribution control inputs with prediction errors orders of magnitude larger than those of our HNN. This reveals the inherent brittleness of black-box models: they often learn superficial correlations rather than the underlying physical laws, making them unreliable when confronted with novel situations.

In contrast, our HNN model embodies the benefit of a strong, physically grounded inductive bias. By being architecturally constrained to adhere to the principles of Hamiltonian mechanics, it sacrifices a marginal degree of interpolative fitting capability for immense robustness and generalizability. Its ability to maintain high-fidelity predictions under chaotic conditions is its defining feature. This is powerfully evidenced by its independent discovery of the AUV's physical symmetries ($M_{22} \approx M_{33}$ and $M_{55} \approx M_{66}$), which is a property learned purely from data rather than being hard-coded in the network architecture. This demonstrates that the model does not simply fit data; it successfully discovers the system's fundamental laws of motion, which is the key to its robust performance.

5.2. Framework Generality and the Path to Real-World Application

It is crucial to distinguish between the implementation choices for our case study and the inherent generality of the proposed framework. The use of a simplified, predominantly diagonal structure for the learned mass and damping matrices was a deliberate engineering choice, which was justified by the well-understood, low-speed dynamics of the torpedo-shaped REMUS 100 AUV. This is not a fundamental limitation. For vehicles with more complex geometries or for high-speed regimes with strong hydrodynamic coupling, our framework can be directly configured to learn full, dense mass and damping matrices by simply adjusting the network output layers. This inherent scalability is a core strength of our methodology.

Furthermore, while this study serves as a crucial proof-of-concept within a controlled simulation environment, we recognize that bridging the "sim-to-real" gap is the ultimate objective. The current work lays the necessary foundation for this transition. Future research will proceed along a clear roadmap to address real-world complexities:

- Data-driven Robustness: We will focus on transfer learning, using the current model pre-trained on extensive simulation data as a powerful prior, and then fine tuning it on smaller, targeted datasets of real-world experimental trajectories. This strategy is expected to drastically reduce the need for expensive and time-consuming physical experiments.
- Environmental Disturbances: The Port-Hamiltonian structure can be extended to include input ports for unmodeled dynamics and external disturbances like ocean currents. Future work will investigate learning these disturbance terms or integrating the model with online estimation algorithms.
- Noise Modeling: To enhance robustness, domain randomization techniques will be applied during training, augmenting the simulation data with realistic sensor noise models (e.g., for IMUs and DVLs) to prepare the model for deployment on physical hardware.

This structured approach, built upon the validated physics-informed foundation of our current work, charts a clear course toward developing highly reliable and deployable dynamics models for real-world maritime systems.

6. Conclusions

In this paper, we introduced a novel physics-informed framework for the high-fidelity, long-term prediction of underwater vehicle dynamics by embedding Port-Hamiltonian mechanics within a neural ordinary differential equation architecture. By structurally separating energy-conserving and dissipative effects and leveraging a symplectic integrator, our model successfully learns the underlying energy landscape of the AUV directly from trajectory data. Our comprehensive experiments demonstrated that the Hamiltonian neural network (HNN) achieves exceptional robustness and accuracy, particularly under complex, chaotic control inputs where conventional black-box models like the TCN catastrophically fail. The quantitative results, revealing a performance gap of orders of magnitude, underscore the critical role of physical priors in ensuring model trustworthiness. While validated in a high-fidelity simulation, this work establishes a crucial baseline for real-world deployment. Our work provides compelling evidence that integrating physical laws is a foundational requirement for overcoming the inherent brittleness of purely data-driven methods, offering a clear path toward developing reliable, generalizable, and safe dynamic models for critical engineering systems.

Author Contributions: Conceptualization, X.J.; methodology, X.J. and Z.L.; software, X.J.; validation, J.L.; formal analysis, X.J. and Z.L.; writing—original draft preparation, X.J.; writing—review and editing, Z.L., J.L. and Y.L.; visualization, X.J. and Z.L.; supervision, Y.L.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research and the APC were funded by the Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai), grant number SML2024SP033.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Nomenclature

This nomenclature provides a list of the essential variables and symbols used throughout the manuscript.

Coordinate Frames and Kinematics	
{n}	Earth-fixed coordinate frame
{b}	Body-fixed coordinate frame
$\boldsymbol{\eta}$	Generalized position/orientation vector in frame n; $\boldsymbol{\eta} = [\boldsymbol{p}^{nT}, \boldsymbol{\Theta}^T]^T$
$\boldsymbol{p}^n$	Position vector (x, y, z) of the vehicle origin in Earth-fixed frame
$\boldsymbol{\Theta}$	Orientation vector (Euler angles) (ϕ, θ, ψ) in Earth-fixed frame
ϕ, θ, ψ	Roll, pitch, and yaw angles
$\boldsymbol{\nu}$	Generalized velocity vector in body-fixed frame; $\boldsymbol{\nu} = [\boldsymbol{v}^T, \boldsymbol{\omega}^T]^T$
$\boldsymbol{v}$	Linear velocity vector (u, v, w) in body-fixed frame
$\boldsymbol{\omega}$	Angular velocity vector (p, q, r) in body-fixed frame
$\boldsymbol{J}(\boldsymbol{\eta})$	Kinematic transformation matrix from frame {b} to frame {n}
Rigid Body and Hydrodynamics	
$\boldsymbol{M}$	System inertia matrix (rigid body + added mass)
$\boldsymbol{C}(\boldsymbol{\nu})$	Coriolis and centripetal matrix (rigid body + added mass)
$\boldsymbol{D}(\boldsymbol{\nu})$	Hydrodynamic damping matrix
$\boldsymbol{g}(\boldsymbol{\eta})$	Vector of gravitational and buoyancy forces and moments
$\boldsymbol{\tau}$	Vector of control inputs (forces and moments from actuators)
Hamiltonian Formulation	
$\boldsymbol{q}$	Generalized coordinates in phase space
$\boldsymbol{p}$	Generalized momentum in phase space; $\boldsymbol{p} = \boldsymbol{M}\boldsymbol{\nu}$
$H(\boldsymbol{q}, \boldsymbol{p})$	Hamiltonian function (total system energy)
$T(\boldsymbol{p})$	Kinetic energy component of the Hamiltonian
$V(\boldsymbol{q})$	Potential energy component of the Hamiltonian
$\mathbf{J}(\boldsymbol{q}, \boldsymbol{p})$	Interconnection matrix (describes conservative effects)
$\mathbf{R}(\boldsymbol{q}, \boldsymbol{p})$	Dissipation matrix (describes dissipative effects)
$\mathbf{G}(\boldsymbol{q}, \boldsymbol{p})$	Control input matrix in the Port-Hamiltonian formulation
SE(3) Representation for Rotation	
$\boldsymbol{R}$	Rotation matrix from frame {b} to frame {n}; an element of SO(3)
$\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3$	Column vectors of the rotation matrix $\boldsymbol{R}$
SO(3)	Special orthogonal group in three dimensions (rotation matrices)
$\mathfrak{so}(3)$	Lie algebra of SO(3) (3×3 skew-symmetric matrices)
SE(3)	Special Euclidean group in three dimensions (rigid-body motions)
$\mathfrak{se}(3)$	Lie algebra of SE(3)
$(\cdot)^\wedge$	Hat operator, mapping a vector to its skew-symmetric matrix form
$(\cdot)^\vee$	Vee operator, the inverse of the hat operator

Appendix A. Derivation of the Port-Hamiltonian AUV Model

This appendix provides a detailed derivation of the Port-Hamiltonian (PH) formulation from the classical 6-DOF underwater vehicle model presented in Section 3.1.

To reformulate the traditional model in PH form, we first define the state variables in phase space. We select the generalized coordinates $\boldsymbol{q}$ to be the vehicle's position and orientation vector, and we select the generalized momentum $\boldsymbol{p}$ as the product of the inertia matrix and the velocity vector:

$$\boldsymbol{q} = \boldsymbol{\eta}, \quad \boldsymbol{p} = M\boldsymbol{\nu} \tag{A1}$$

With these definitions, the kinematic equation (Equation (12)) can be expressed in terms of the momentum:

$$\dot{\boldsymbol{q}} = J(\boldsymbol{q})\boldsymbol{\nu} = J(\boldsymbol{q})M^{-1}\boldsymbol{p} \tag{A2}$$

Next, we reformulate the dynamic equation (Equation (16)). By substituting $\boldsymbol{\nu} = M^{-1}\boldsymbol{p}$ and assuming the inertia matrix M is constant (a common assumption for rigid-body vehicles), we can express the rate of change of momentum $\dot{\boldsymbol{p}} = M\dot{\boldsymbol{\nu}}$ as

$$\dot{\boldsymbol{p}} = -C(M^{-1}\boldsymbol{p})M^{-1}\boldsymbol{p} - D(M^{-1}\boldsymbol{p})M^{-1}\boldsymbol{p} - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau} \tag{A3}$$

The total energy of the system is described by the Hamiltonian function $H(\boldsymbol{q}, \boldsymbol{p})$, which is the sum of the kinetic energy $T(\boldsymbol{p})$ and potential energy $V(\boldsymbol{q})$:

$$H(\boldsymbol{q}, \boldsymbol{p}) = T(\boldsymbol{p}) + V(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{p}^T M^{-1}\boldsymbol{p} + V(\boldsymbol{q}) \tag{A4}$$

The gradients of the Hamiltonian with respect to the generalized coordinates and momentum are

$$\frac{\partial H}{\partial \boldsymbol{q}} = \frac{\partial V}{\partial \boldsymbol{q}}, \quad \frac{\partial H}{\partial \boldsymbol{p}} = M^{-1}\boldsymbol{p} = \boldsymbol{\nu} \tag{A5}$$

Using these components, we can now write the full system dynamics in the standard Port-Hamiltonian form:

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{bmatrix} = (\mathbf{J}(\boldsymbol{q}, \boldsymbol{p}) - \mathbf{R}(\boldsymbol{q}, \boldsymbol{p})) \begin{bmatrix} \frac{\partial H}{\partial \boldsymbol{q}} \\ \frac{\partial H}{\partial \boldsymbol{p}} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \boldsymbol{\tau} \tag{A6}$$

where the skew-symmetric interconnection matrix $\mathbf{J}$ (representing conservative effects) and the positive semi-definite dissipation matrix $\mathbf{R}$ (representing dissipative effects) are defined as

$$\mathbf{J} = \begin{bmatrix} 0 & J(\boldsymbol{q}) \\ -J^T(\boldsymbol{q}) & -C(M^{-1}\boldsymbol{p}) \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 0 & 0 \\ 0 & D(M^{-1}\boldsymbol{p}) \end{bmatrix} \tag{A7}$$

For this formulation to be consistent, the potential energy function $V(\boldsymbol{q})$ must be chosen such that its gradient correctly represents the gravitational and buoyancy forces $\boldsymbol{g}(\boldsymbol{q})$. This relationship is given by

$$-\boldsymbol{J}^T(\boldsymbol{q})\frac{\partial V}{\partial \boldsymbol{q}} = \boldsymbol{g}(\boldsymbol{q}) \tag{A8}$$

While this equation may not have a general analytical solution, an analytical solution can often be derived for neutrally buoyant vehicles with coincident centers of gravity and buoyancy in the lateral plane. This is a reasonable assumption for slender underwater vehicles, which are typically trimmed to be neutrally or slightly positively buoyant. For more complex vehicle geometries, numerical methods may be required to find a suitable potential energy function.

References

1. Lv, B.; Huang, B.; Ming, T.; Peng, L. Identification method of hydrodynamic coefficients of underwater vehicles based on Kalman filter. *J. Wuhan Univ. Technol. Transp. Sci. Eng.* **2021**, *45*, 464–469.
2. Harris, Z.J.; Paine, T.M.; Whitcomb, L.L. Preliminary Evaluation of Null-Space Dynamic Process Model Identification with Application to Cooperative Navigation of Underwater Vehicles. In Proceedings of the 31st IROS Conference, Madrid, Spain, 1–5 October 2018; pp. 3453–3459. [CrossRef]
3. Chu, S.; Mao, Y.; Dong, Z.; Yang, X.; Huang, C. Parameter identification of unmanned boat maneuvering response model based on UKF. *J. Wuhan Univ. Technol. Transp. Sci. Eng.* **2019**, *43*, 947–950.
4. Ramirez, W.A.; Kocijan, J.; Leong, Z.Q.; Nguyen, H.D.; Jayasinghe, S.G. Dynamic System Identification of Underwater Vehicles Using Multi-Output Gaussian Processes. *Int. J. Autom. Comput.* **2021**, *18*, 681–693. [CrossRef]
5. Shafiei, M.; Binazadeh, T. Application of neural network and genetic algorithm in identification of a model of a variable mass underwater vehicle. *Ocean Eng.* **2015**, *96*, 173–180. [CrossRef]

6. Zhao, R.; Xie, H.; Qiao, B. A method for identifying hydrodynamic parameters of underwater vehicles based on experimental data. *J. Under. Unmanned Syst.* **2018**, *26*, 335–341.
7. McCarter, B. Experimental Evaluation of Viscous Hydrodynamic Force Models for Autonomous Underwater Vehicles. Ph.D. Thesis, Virginia Tech, Blacksburg, VA, USA, 2014.
8. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 29th CVPR Conference, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
9. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
10. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]
11. Lu, L.; Jin, P.; Pang, G.; Zhang, Z.; Karniadakis, G.E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **2021**, *3*, 218–229. [CrossRef]
12. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th NeurIPS Conference, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105. [CrossRef]
13. Bronstein, M.M.; Bruna, J.; LeCun, Y.; Szlam, A.; Vandergheynst, P. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process Mag.* **2017**, *34*, 18–42. [CrossRef]
14. Cohen, T.; Weiler, M.; Kicanaoglu, B.; Welling, M. Gauge Equivariant Convolutional Networks and the Icosahedral CNN. In Proceedings of the 36th ICML Conference, Long Beach, CA, USA, 9–15 June 2019; pp. 1321–1330.
15. Veeling, B.S.; Linmans, J.; Winkens, J.; Cohen, T.; Welling, M. Rotation Equivariant CNNs for Digital Pathology. In *Medical Image Computing and Computer Assisted Intervention—MICCAI 2018: 21th MICCAI Conference, Granada, Spain, 16–20 September 2018*; Springer: Cham, Switzerland, 2018; pp. 210–218. [CrossRef]
16. Sirignano, J.; Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **2018**, *375*, 1339–1364. [CrossRef]
17. Raissi, M.; Perdikaris, P.; Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
18. Geneva, N.; Zabaras, N. Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *J. Comput. Phys.* **2020**, *403*, 109056. [CrossRef]
19. Chen, R.T.Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D. Neural Ordinary Differential Equations. In Proceedings of the 32nd NeurIPS Conference (NIPS'18), Montréal, QC, Canada, 2–8 December 2018; pp. 6572–6583.
20. Greydanus, S.; Dzamba, M.; Yosinski, J. Hamiltonian Neural Networks. In Proceedings of the 33rd NeurIPS Conference, Vancouver, BC, Canada, 8–14 December 2019; pp. 15353–15363.
21. Sosanya, A.; Greydanus, S. Dissipative hamiltonian neural networks: Learning dissipative and conservative dynamics separately. *arXiv* **2022**, arXiv:2201.10085. [CrossRef]
22. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. In Proceedings of the 31st NeurIPS Conference (NIPS'17), Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
23. Zhong, Y.D.; Dey, B.; Chakraborty, A. Symplectic ode-net: Learning hamiltonian dynamics with control. In Proceedings of the 8th ICLR Conference, Addis Ababa, Ethiopia, 26–30 April 2020; pp. 1–17.
24. Howse, J.W.; Abdallah, C.T.; Heileman, G.L. Gradient and Hamiltonian Dynamics Applied to Learning in Neural Networks. In Proceedings of the 8th NeurIPS Conference (NIPS'95), Cambridge, MA, USA, 27 November–2 December 1995; pp. 274–280.
25. Seung, H.S.; Richardson, T.J.; Lagarias, J.C.; Hopfield, J.J. Minimax and Hamiltonian Dynamics of Excitatory-Inhibitory Networks. In Proceedings of the 10th NeurIPS Conference (NIPS'97), Denver CO, USA, 1 December 1997; pp. 329–335.
26. Leabourne, K.; Rock, S. Model development of an underwater manipulator for coordinated arm-vehicle control. In Proceedings of the 11th OCEANS Conference, Nice, France, 28 September–1 October 1998; Volume 2, pp. 941–946. [CrossRef]
27. McLain, T.W.; Rock, S.M. Development and Experimental Validation of an Underwater Manipulator Hydrodynamic Model. *Int. J. Robot Res.* **1998**, *17*, 748–759. [CrossRef]
28. McLain, T.W.; Rock, S.M.; Lee, M.J. Experiments in the coordinated control of an underwater arm/vehicle system. *Auton. Robot.* **1996**, *3*, 213–232. [CrossRef]
29. Fossen, T.I. *Handbook of Marine Craft Hydrodynamics and Motion Control*, 1st ed.; Wiley: Hoboken, NJ, USA, 2011. [CrossRef]
30. Ortega, R.; Van Der Schaft, A.; Maschke, B.; Escobar, G. Interconnection and damping assignment passivity-based control of port-controlled Hamiltonian systems. *Automatica* **2002**, *38*, 585–596. [CrossRef]
31. Jia, Z.; Qiao, L.; Zhang, W. Adaptive tracking control of unmanned underwater vehicles with compensation for external perturbations and uncertainties using Port-Hamiltonian theory. *Ocean Eng.* **2020**, *209*, 107402. [CrossRef]
32. Zhong, Y.D.; Dey, B.; Chakraborty, A. Dissipative symoden: Encoding hamiltonian dynamics with dissipation and control into deep learning. *arXiv* **2020**, arXiv:2002.08860. [CrossRef]

33. Duong, T.; Atanasov, N. Hamiltonian-based Neural ODE Networks on the SE(3) Manifold For Dynamics Learning and Control. In Proceedings of the 17th RSS Conference Robotics: Science and Systems Foundation, Virtually, 12–16 July 2021; pp. 1–13. [CrossRef]
34. Duong, T.; Atanasov, N. Adaptive Control of SE(3) Hamiltonian Dynamics With Learned Disturbance Features. *IEEE Control Syst. Lett.* **2022**, *6*, 2773–2778. [CrossRef]
35. Chen, Z.; Zhang, J.; Arjovsky, M.; Bottou, L. Symplectic recurrent neural networks. In Proceedings of the 8th ICLR Conference, Addis Ababa, Ethiopia, 26–30 April 2020; pp. 1–23.
36. Saviolo, A.; Li, G.; Loianno, G. Physics-Inspired Temporal Learning of Quadrotor Dynamics for Accurate Model Predictive Trajectory Tracking. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10256–10263. [CrossRef]