Poisson Hamiltonian Neural Networks: Structure-Preserving Learning of Dynamical Systems

Adérito Araújo[1][0000-0002-9873-5974], Gonçalo Inocêncio Oliveira[1][0000-0001-9312-2004], and João Nuno Mestre[1][0000-0003-1217-9872]

CMUC, Department of Mathematics, University of Coimbra, Portugal
g.inoc.oliveira@gmail.com

Abstract. In this work, we introduce Poisson Hamiltonian Neural Networks (PHNNs) as an extension of Hamiltonian Neural Networks to better capture the dynamics of Poisson-Hamiltonian systems. By incorporating structure-preserving numerical methods, PHNNs can learn a wider range of dynamical systems beyond traditional symplectic models. We explored different training strategies, comparing Explicit Euler (EE) and Poisson-Hamiltonian Integrators (PHI). Our results showed that, while Euler-trained models offer better short-term accuracy, PHI-trained models stand out for their long-term stability and preservation of geometric structures. A hybrid approach – training with EE and testing with PHI – proved to be the best balance between accuracy and stability. These results highlight the potential of combining machine learning with geometric numerical methods to model complex dynamic systems without the need for explicit governing equations.

Keywords: Artificial Neural Networks · Hamiltonian Systems · Poisson Integrators.

1 Introduction

Machine learning has become a powerful tool for modelling and predicting complex dynamical systems. Among various approaches, Artificial Neural Networks have demonstrated success in applications such as image classification [10], facial expression recognition [8] and medical diagnosis [9]. Neural networks have also found an extensive application in understanding and predicting the outcomes of dynamic systems such as fluid dynamics [15], weather patterns [1], and space weather [6].

Hamiltonian Neural Networks (HNNs) [11] have emerged as a method to learn physical systems while preserving energy conservation laws. Poisson-Hamiltonian systems, which generalize Hamiltonian mechanics, offer a broader framework for studying dynamical systems beyond traditional symplectic structures. In this paper, we introduce Poisson Hamiltonian Neural Networks (PHNNs), an extension of HNNs that enables the learning and prediction of Poisson-Hamiltonian

[v1] Fri, 26 Jun 2026

systems while preserving their underlying geometric structure. Unlike standard neural network approaches that approximate the vector field directly, PHNNs leverage the Poisson structure to ensure physically consistent dynamics. Furthermore, we integrate Poisson-Hamiltonian Integrators (PHIs) into the learning process, showing their effectiveness in improving long-term stability and energy conservation compared to explicit numerical methods. Through extensive experiments on Lotka-Volterra systems [5] and Rigid Body Rotation [14], we show that PHNNs outperform baseline methods in preserving key invariants of the system. This framework paves the way for more robust and interpretable machine learning models for complex dynamical systems.

2 Literature Survey

The modeling of physical systems through machine learning has attracted considerable attention in recent years, particularly with the advent of methods that incorporate prior knowledge of the system's structure into learning frameworks. This includes a class of neural networks known as Hamiltonian Neural Networks (HNNs), introduced by Greydanus et al. [11], which learn the Hamiltonian of a system directly from data, thereby preserving energy conservation and symplectic structure.

Hamiltonian systems are classically formulated on symplectic manifolds, which are even-dimensional and equipped with a non-degenerate, closed 2-form. However, many real-world systems possess a more general geometric structure – Poisson manifolds – where the non-degeneracy condition is relaxed. Poisson geometry provides a broader framework to describe dynamics, including systems like the Lotka-Volterra equations and rigid-body rotations, which do not naturally fit into symplectic geometry.

A Poisson structure [5] on $\mathbb{R}^n$ is a bilinear operation (also called a Poisson bracket) $\{\cdot,\cdot\} : C^\infty(\mathbb{R}^n) \times C^\infty(\mathbb{R}^n) \to C^\infty(\mathbb{R}^n)$ which satisfies, for all $f, g, h \in C^\infty(\mathbb{R}^n)$:

(i) $\{f, g\} = -\{g, f\}$ (skew-symmetry);
(ii) $\{f, \{g, h\}\} + \{g, \{h, f\}\} + \{h, \{f, g\}\} = 0$ (Jacobi identity);
(iii) $\{f, gh\} = g\{f, h\} + h\{f, g\}$ (Leibniz identity).

The first two conditions mean that $\{\cdot,\cdot\}$ is a Lie bracket on $C^\infty(\mathbb{R}^n)$, the space of all smooth functions from $\mathbb{R}^n$ to $\mathbb{R}$, defining a Lie algebra $(C^\infty(\mathbb{R}^n), \{\cdot,\cdot\})$.This generalizes the familiar symplectic structure and appears in a variety of important examples:

(i) Symplectic structures: The canonical structure of classical mechanics, $\{p_i, q^j\} = \delta_{ij}$ and $\{p_i, p_j\} = \{q^j, q^j\} = 0,\ i, j = 1, \ldots, n$;
(ii) Constant Poisson structures: $\{x^i, x^j\} = c^{ij},\ c^{ij} \in \mathbb{R}$;
(iii) Linear Poisson structure: $\{x^i, x^j\} = \sum_{k=1}^{n} c_k^{ij} x^k$, where the constants c_k^{ij} are structure constants of a Lie algebra;
(iv) Quadratic Poisson structures (Lotka-Volterra type): $\{x^i, x^j\} = a^{ij} x^i x^j$, where (a_{ij}) is a skew-symmetric matrix.

Given coordinates $x = (x^1, \ldots, x^n)$ on $\mathbb{R}^n$, a Poisson structure can be expressed as

$$\{f, g\}(x) = \sum_{i,j=1}^{n} \pi^{ij}(x) \frac{\partial f}{\partial x^i} \frac{\partial g}{\partial x^j}, \tag{1}$$

with $\pi^{ij}(x) \in C^\infty(\mathbb{R}^n)$ given by $\pi^{ij}(x) = \{x^i, x^j\}(x)$. Using the skew-symmetric matrix $\Pi(x) = \left(\pi^{ij}(x)\right)_{i,j}$ we can translate (1) into matrix multiplication

$$\{f, g\}(x) = \nabla f(x) \Pi(x) \nabla g(x). \tag{2}$$

By definition, given functions $f, g \in C^\infty(\mathbb{R}^n)$, $\{f, g\}$ is another function. Derivations of the algebra of smooth functions are vector fields, so the Leibniz identity ensures that, given a function $H \in C^\infty(\mathbb{R}^n)$, we can define a vector field $X_H : C^\infty(\mathbb{R}^n) \to C^\infty(\mathbb{R}^n)$, by $X_H f = \{f, H\}$. We call H a Hamiltonian function (playing the role of energy of a system) and X_H its associated Hamiltonian vector field. The corresponding Poisson-Hamiltonian system is

$$\dot{x} = \Pi(x) \nabla H(x). \tag{3}$$

Any symplectic Hamiltonian system is, in fact, also a Poisson system. For instance, if $x = (q, p) \in \mathbb{R}^{2n}$, the equations of motion can be written as

$$\dot{x} = J \nabla H(x), \qquad J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}, \tag{4}$$

with H as the Hamiltonian and I_n the $n \times n$ identity matrix. Note that, symplectic Hamiltonian systems are only defined on spaces of even dimension, unlike Poisson-Hamiltonian systems which can be defined in any dimension.

Numerical integrators that preserve the geometric properties of differential equations are essential for long-term simulations of dynamical systems. Classical explicit integrators like Euler's method can introduce numerical dissipation or instability, especially over long times. In contrast, geometric integrators, such as symplectic and Poisson integrators, maintain invariants of the system, such as volume or energy, leading to more physically consistent simulations [12].

For Poisson systems, preserving both the Poisson structure and the Hamiltonian is more involved due to possible degeneracies in the structure. The Poisson-Hamiltonian Integrator (PHI), based on the theory of symplectic groupoids and bi-realizations, addresses this issue by performing integration in an extended symplectic space and projecting results back onto the Poisson manifold [4].

The integration of machine learning with geometric structures led to the emergence of Hamiltonian Neural Networks (HNNs), which approximate the Hamiltonian H rather than the vector field $\dot{x}$ directly. This guarantees that the learned dynamics respect conservation laws inherent to physical systems [11].

Extending this idea, Poisson Hamiltonian Neural Networks (PHNNs) are designed to operate on systems with Poisson structure. PHNNs leverage the mathematical formulation of Poisson brackets and can be trained using structure-preserving integrators such as PHI. This approach allows for the accurate modeling of a wider range of dynamical systems without the need for explicitly known differential equations.

Other notable contributions include Symplectic Recurrent Neural Networks [2], which further develop structure-preserving learning, and Lie-Poisson Neural Networks (LPNets) [7], which extend these ideas to systems with symmetry and Lie group structure.

These advancements have enabled the application of geometric machine learning techniques in diverse areas such as fluid dynamics [15], ionospheric space weather prediction [6], medical imaging [9], and remote sensing [10]. They also open up possibilities for data-driven discovery in control systems, quantum mechanics, and plasma physics, where governing equations may be partially known or entirely unavailable.

The present work builds upon this foundation by proposing PHNNs, evaluating different training and testing strategies that combine explicit integrators, such as the Euler method, with structure-preserving schemes known as Poisson-Hamiltonian integrators. We validate the approach on benchmark systems, including the pendulum, Lotka-Volterra models, and rigid-body rotation.

3 Methodology

There are several numerical methods, both classical and geometric integrators, that can solve Poisson-Hamiltonian system numerically. However, they need to explicitly know the differential equation in order to act on it. On the other hand, deep learning tools can operate using only data from the system, which is an advantage if we do not know the differential equation behind the data generation process. Recent advances have integrated deep learning with Hamiltonian mechanics, leading to the development of Hamiltonian Neural Networks (HNNs) [11]. Instead of directly approximating the vector field, HNNs approximate the Hamiltonian itself, ensuring that the learned dynamics respect the conservation laws.

We propose Poisson Hamiltonian Neural Networks (PHNNs), which extend the concept of HNNs to systems described by Poisson geometry. Given a Poisson-Hamiltonian system defined by (3) is a skew-symmetric matrix representing the Poisson structure, PHNNs learn an approximation $H_\theta(x)$ to the Hamiltonian. The training process consists of the following steps:

1. Generating Trajectories: Initial points $x_{0,k}$ are randomly sampled, and corresponding trajectories $\{x_k\}$ are computed using numerical integration.
2. Constructing the Training Set: The values of the vector field at sampled points are used to form the dataset $D = \{(x_k, \dot{x}_k) \,|\, \dot{x}_k = \Pi(x_k)\nabla H(x_k)\}$.
3. Neural Network Training: A fully connected neural network is used to approximate $H_\theta(x)$. The loss function is defined as:

$$\mathcal{L}(x, \dot{x}) = \frac{1}{N} \sum_{k=1}^{N} \|\dot{x}_k - \Pi(x_k)\nabla H_\theta(x_k)\|^2 .$$

We observe that these neural networks can be used to approximate the Hamiltonian based on the values of the vector field. However, in many cases, the vector

field is not explicitly available, making numerical integration necessary. In [2], a geometric integrator is employed during the training phase to compute the predicted trajectory and compare it with the true solution. Similarly, since we only have access to an approximate vector field at each learning iteration, we can utilize a (Poisson) geometric integrator to compare the predicted and actual trajectories.

We will now explain how combine PHNN and numerical integrators. We denote H_θ as the output of the neural network, thus gaining access to the approximate vector field by calculating $\Pi(x_i)\nabla H_\theta(x_i)$. Now, we will use a numerical integrator to compute the trajectory predicted by the model. However, we will only take one step of this integration, which means $x_k^* = integrator(x_{k-1}, \Delta t)$. This corresponds to what is called the single H-NET in [2]. The loss function in this context becomes

$$\mathcal{L}(x, x^*) = \frac{1}{N}\sum_{i=k}^{N} \|x_k - x_k^*\|^2,$$

where x_k^* is the result of a single integration step using the chosen method. This formulation allows for the evaluation of both short-term accuracy and long-term stability of PHNNs under different integration strategies. Selecting an appropriate numerical integrator is critical for accurately simulating and analysing complex dynamical systems with underlying Poisson structure.

The theory of geometric integrators is well established for non-degenerate Poisson structures, more precisely, for symplectic systems [12]. In this context, preserving the symplectic structure alone ensures that the system remains on the level sets of the Hamiltonian, thereby conserving energy. However, for general Poisson systems, which may be degenerate, preserving the Poisson structure is not sufficient to guarantee energy conservation. In such cases, it becomes necessary to use an integrator that preserves both the Poisson structure and the Hamiltonian, at least to a certain order of accuracy. A numerical scheme that fulfills these two conditions – the Poisson-Hamiltonian Integrator (PHI) – was introduced in [4]. This integrator is based on the theory of symplectic groupoids and their role in integrating Poisson manifolds.

Poisson structures can be defined on any smooth manifold [5] and, in local coordinates, are represented by a skew-symmetric matrix of functions Π. A pair (M, Π), where M is a smooth manifold and Π a Poisson structure, defines what is known as a Poisson manifold.

The PHI method exploits a fundamental result in Poisson geometry: any Poisson manifold – despite potential degeneracies – can be embedded into a higher-dimensional symplectic manifold that retains all the essential features of the original structure, such as its symplectic foliation. This construction is known as a symplectic realization (see [5, Chapter 6]).

Such realizations always exist and can be equipped with a local symplectic groupoid structure. A symplectic groupoid includes two projection maps onto the original Poisson manifold: the *source map* α and the *target map* β. Together, these form a *bi-realization* of the Poisson manifold. Importantly, α is a Poisson

map, β is an anti-Poisson map, and their composition $\beta \circ \alpha^{-1}$ defines a Poisson diffeomorphism.

Within this groupoid framework, for any Hamiltonian function $H \in C^\infty(M)$, one can define a family of *Lagrangian bisections* $(L_t)_t$, which are submanifolds where both α and β restrict to local diffeomorphisms onto M. These bisections are generated by solutions of the associated Hamilton-Jacobi equation.

Given a time step Δt, the PHI method proceeds as follows:

1. Choose the Lagrangian bisection $L_{\Delta t}$ associated with that time step;
2. For a current state x_k, find a point $y_k \in L_{\Delta t}$ such that $\alpha(y_k) = x_k$;
3. Compute the next state as $x_{k+1} = \beta(y_k)$.

This process is visually illustrated in Figure 1.

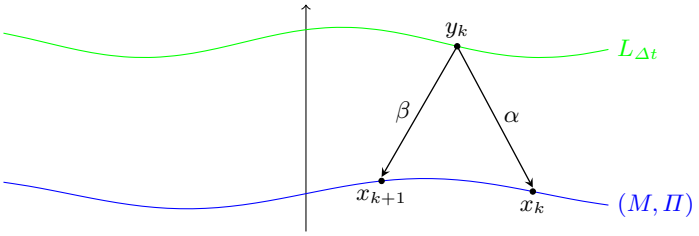


Fig. 1: Visualization of how the PHI method iterates.

This approach guarantees the preservation of the Poisson structure and controlled conservation of the Hamiltonian while ensuring that the numerical flow remains within the symplectic leaf of the Hamiltonian system [4]. For a more detailed explanation of the method, see [3].

4 Results and discussion

In this work, we use several benchmark dynamical systems to illustrate how Poisson structures govern physical behaviour. These examples serve to demonstrate the versatility and effectiveness of the proposed algorithms in learning dynamics that preserve the underlying geometric structure.

A simple but illustrative case of a symplectic Hamiltonian system (4) is the pendulum, with $(q, p) \in \mathbb{R}^2$ and Hamiltonian

$$H(q, p) = \frac{p^2}{2ml^2} + mgl(1 - \cos(q)), \tag{5}$$

where m is the mass, l the length, and g the gravitational constant.

The Lotka-Volterra system, which models the population dynamics of interacting species, provides a more complex example with a quadratic Poisson

structure

$$\{x^i, x^j\} = \begin{cases} x^i x^j, & i < j, \\ -x^i x^j, & i > j, \end{cases} \quad H(x) = \sum_{i=1}^{n} x^i - \lambda_i \log(x^i), \tag{6}$$

where the λ_i are system parameters. The dynamics are no longer symplectic but still follow a conservative, structure-preserving evolution.

A classical system with a linear Poisson structure is the rigid-body rotation, where the evolution equations are expressed using a skew-symmetric matrix:

$$\Pi(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}, \quad H(x) = \frac{1}{2}\left[(I_2 + I_3)x_1^2 + (I_1 + I_3)x_2^2 + (I_1 + I_2)x_3^2\right], \tag{7}$$

with I_1, I_2, I_3 representing the principal moments of inertia.

We will create a PHNN for 4 examples: (i) the symplectic pendulum (4)–(5); (ii) the quadratic Lotka-Volterra (6) in 2 dimensions; (iii) the quadratic Lotka-Volterra (6) in 3 dimensions; (iv) the linear rigid-body problem (7).

4.1 Training with vector field

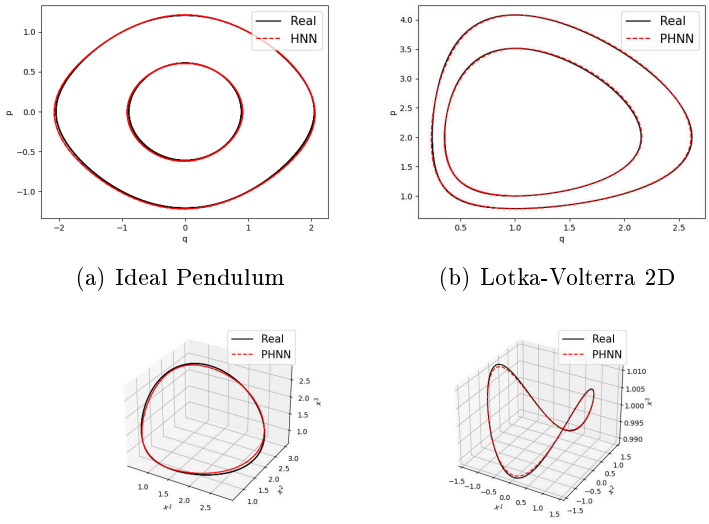
For training we use a PHNN with four layers, both hidden layers will have 200 neurons each, and the activation functions are $\tanh(x)$. The number of epochs will be 2000, and for each iteration the *training set* will have $N = 2250$ points, added Gaussian noise with $\sigma^2 = 0.1$ to simulate real data. The optimizer used is Adam updater [13] with a learning rate of 10^{-3}.

Ideal Pendulum. Consider the ideal pendulum, $m = l = g = 1$ in the system (4)–(5). In this case, the PHNN will be an HNN originally. In Figure 2 we can see the difference between the real trajectory and the one predicted by HNN, both essentially coincide, with initial points $(1, 1)$ and $(0.5, 0.5)$.

Lotka-Volterra 2D. Consider the Poisson structure (6) in two dimensions with $\lambda_1 = 1$ and $\lambda_2 = 2$. In this case, we are only interested in the points in $\mathbb{R}^2_+$ and the dynamics occurs around the stationary point $(1, 2)$. So, we will randomly generate the initial points in the square $[0, 2]\times[1, 3]$ and compute their trajectory. After training, the results of predicted trajectory and the real one for the point $x_0 = (1, 1)$ and $x_0 = (0.8, 0.8)$ for time $t \in [0, 2\pi]$ can be seen in Figure 2. We can see that the predicted solution still coincides with the real solution, as in the symplectic case, even for initial values outside of the generating square.

Lotka-Volterra 3D. Consider now the quadratic Poisson structure (6) in three dimensions with $\lambda_1 = 1$, $\lambda_2 = 2$ and $\lambda_3 = 1$. The initial points will take place in $[0, 2] \times [1, 3] \times [0, 2]$. After training, let us compute the trajectory predicted by the PHNN for the initial point $(2, 3, 2)$ and time span $[0, 2\pi]$. In Figure 2 we can see that the PHNN still works in odd dimensions with accurate predictions in a neighbourhood of the *training set*. The predicted solution stays very close to the real solution, even with the initial point at the boundary of the training set.

Rigid Body Rotation. Consider the rigid-body rotation in (7) with inertia values of $I_1 = 1$, $I_2 = \pi$ and $I_3 = 100$. In this case, the training set will have $N = 2500$ points, with the initial points generated in the cube $[0, 1]^3$ with time step $\Delta t = \frac{1}{1386}$ and Gaussian noise $\sigma^2 = 0.1$. After training we compute the predicted trajectory for the initial point $(1, 1, 1)$ and for time span $[0, 1/14]$ that can be seen in Figure 2. Again, the predicted solution by PHNN is still close to the real solution for a point at the boundary of the initial points.



(a) Ideal Pendulum (b) Lotka-Volterra 2D

(c) Lotka-Volterra 3D (d) Rigid Body Rotation

Fig. 2: Comparison between the real trajectory and the predicted one by PHNN.

We can see in Figure 2 that the extension of HNN to PHNN works with high accuracy in the predicted trajectories. In the four experiments, the predicted trajectories overlaps with the real solutions.

4.2 Training with integrators

We will now combine PHNN and numerical integrators and compare the effectiveness of two integrators of order 1: (i) Explicit Euler (EE): $x_{k+1} = x_k + \Delta t \Pi(x) \nabla H(x_k)$; (ii) Poisson-Hamiltonian Integrator (PHI):

$$\begin{cases} \text{Find } y_k \text{ such that } \alpha(y_k, \Delta t \nabla H(y_k)) = x_k, \\ \text{Compute } x_{k+1} = \beta(y_k, \Delta t \nabla H(y_k)). \end{cases} \tag{8}$$

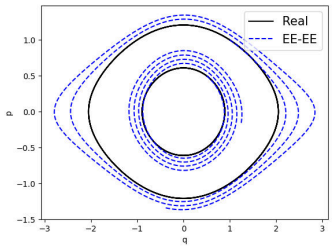
Similarly to what was done in [2], we analyze the impact of numerical integration on training by considering different combinations: EE-EE, EE method

for both training and testing; EE-PHI, EE for training, PHI for testing; PHI-EE, PHI for training, EE for testing; PHI-PHI, PHI for both training and testing. Each model is trained for different Poisson systems: Pendulum; Lotka-Volterra system (2D and 3D); Rigid-Body Rotation.

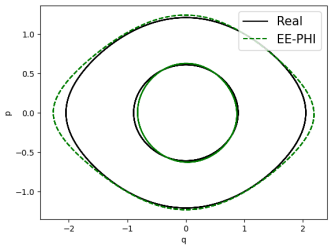
We will employ a neural network with 4 layers, both hidden containing 256 neurons, and output size set to 2. The number of epochs will be 2000, and for each iteration, the training set will consist of 2250 randomly generated points as explained previously, with Gaussian noise added with variance $\sigma^2 = 0.1$. The optimizer will be Adam updater [13] with a learning rate of 10^{-3}.

Ideal Pendulum. For a symplectic structure, one bi-realization takes the form

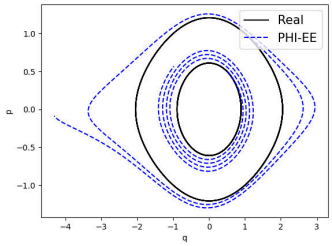
$$\begin{cases} \alpha(q,p,\xi_q,\xi_p) = (q - \frac{1}{2}\xi_p, p + \frac{1}{2}\xi_q), \\ \beta(q,p,\xi_q,\xi_p) = (q + \frac{1}{2}\xi_p, p - \frac{1}{2}\xi_q). \end{cases}$$



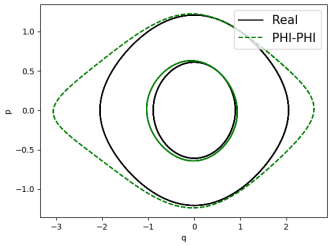
(a) EE-EE model



(b) EE-PHI model



(c) PHI-EE model



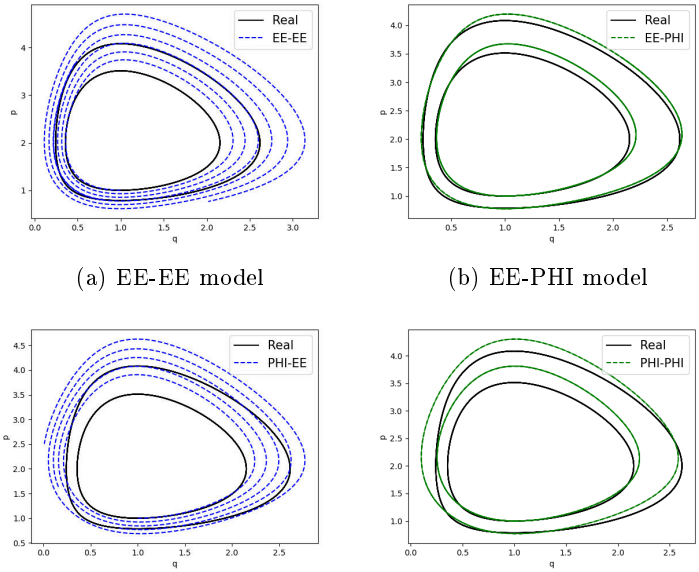
(d) PHI-PHI model

Fig. 3: The PHINN for Ideal Pendulum with initial points $[1, 1]$ and $[0.5, 0.5]$.

We can see in Figure 3 that when we train with Euler method the neural network seems to predict the Hamiltonian with more accuracy. This could be due to the PHI method having an implicit step, so the neural network cannot learn with precision, but when we use PHI in the test phase, the trajectories are stable.

Lotka-Volterra 2D. For Lotka-Volterra in 2 dimensions we have the following bi-realization

$$\begin{cases} \alpha(x,\xi) = (x_1 e^{\frac{1}{2}x_2\xi_2}, x_2 e^{-\frac{1}{2}x_1\xi_1}), \\ \beta(x,\xi) = (x_1 e^{-\frac{1}{2}x_2\xi_2}, x_2 e^{\frac{1}{2}x_1\xi_1}). \end{cases}$$



(a) EE-EE model (b) EE-PHI model

(c) PHI-EE model (d) PHI-PHI model

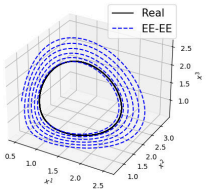
Fig. 4: The PHINN for Lotka-Volterra 2D with initial points $[1, 1]$ and $[0.8, 0.8]$.

In Figure 4 we can see that training with Euler and using PHI in the test phase has better results while integrating with Euler (in trained with Euler or PHI) the energy isn't preserved.

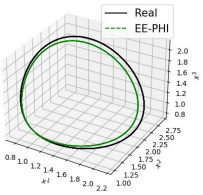
Lotka-Volterra 3D. For Lotka-Volterra in 3 dimensions, the bi-realization is

$$\begin{cases} \alpha(x,\xi) = (x_1 e^{\frac{1}{2}(x_2\xi_2+x_3\xi_3)}, x_2 e^{\frac{1}{2}(x_3\xi_3-x_1\xi_1)}, x_3 e^{-\frac{1}{2}(x_1\xi_1+x_2\xi_2)}), \\ \beta(x,\xi) = (x_1 e^{-\frac{1}{2}(x_2\xi_2+x_3\xi_3)}, x_2 e^{\frac{1}{2}(x_1\xi_1-x_3\xi_3)}, x_3 e^{\frac{1}{2}(x_1\xi_1+x_2\xi_2)}). \end{cases}$$

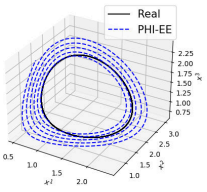
We can see in Figure 5 that when we train with Euler, the predicted trajectory is more accurate than when we use the PHI method, although the difference is not very large. This may suggest that maybe using a simple or explicit numerical method in the training phase has more benefits instead of using an implicit method. However, using PHI in the testing phase has more stability.



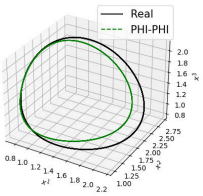
(a) EE-EE model



(b) EE-PHI model



(c) PHI-EE model



(d) PHI-PHI model

Fig. 5: The PHINN for Lotka-Volterra 3D with $x_0 = [1, 1, 1]$.

Rigid Body Rotation. In this case, we will not directly consider a bi-realization of the Poisson structure, but first we will move from $\mathbb{R}^3$ to $so(3)$, via isomorphism, and then consider the bi-realization on the latter space. The isomorphism is $j : \mathbb{R}^3 \xrightarrow{\sim} so(3)$, $j(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & x_1 \\ -x_2 & -x_1 & 0 \end{bmatrix}$ and the bi-realization is

$$\begin{cases} \alpha : so(3) \times so(3) \to so(3) : (A, X) \to (I + \frac{A}{4}).X.(I - \frac{A}{4}), \\ \beta : so(3) \times so(3) \to so(3) : (A, X) \to (I - \frac{A}{4}).X.(I + \frac{A}{4}). \end{cases}$$

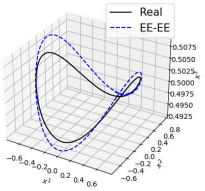
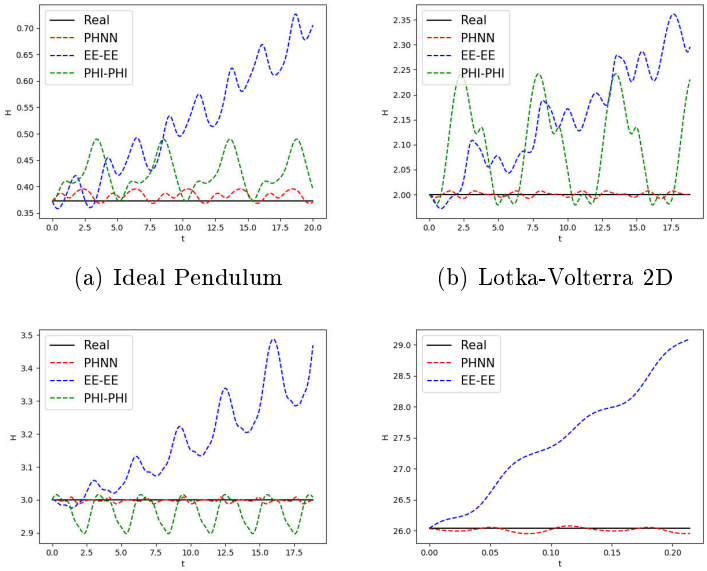


Fig. 6: The PHINN for Rigid Body Rotation with $x_0 = [0.5, 0.5, 0.5]$.

Here the model was trained with 3000 epochs and without noise, that can be seen in Figure 6. The model does not work with the Poisson integrator because in the implicit step, the method cannot find y_k such that $\alpha(y_k, \Delta t \nabla H_\theta(y_k)) = x_k$, and so we cannot iterate the model. Here we need another approach, maybe using a neural network specialized in Lie-Poisson systems as [7].

The Hamiltonian evolution along predicted trajectories was analysed for each system (Figure 7). The PHI-PHI model achieved the best energy conservation property, whereas the EE-EE model introduced accumulating energy errors. Euler-trained models showed small but growing deviations over time. PHI-trained models provided better long-term stability, but introduced implicit errors affected precision. Hybrid training (EE-PHI) achieved the best balance between accuracy and stability.

The choice of numerical integrator significantly influences learning and generalization in PHNNs. Euler-based training yields better short-term accuracy, while PHI-based testing ensures long-term stability. Future research will focus on adaptive integrators and hybrid architectures to enhance performance.



(a) Ideal Pendulum (b) Lotka-Volterra 2D

(c) Lotka-Volterra 3D (d) Rigid Body Rotation

Fig. 7: The value of the real Hamiltonian in each trajectory for each problem.

5 Conclusion

We introduced PHNNs as an extension of HNNs for Poisson-Hamiltonian systems. Using a structure-preserving numerical scheme that preserves the structure and ensures the conservation of both the Poisson and Hamiltonian structures, PHNNs model a broader range of dynamical systems beyond traditional symplectic formulations.

Our study compared different methodologies by integrating PHNNs with Explicit Euler (EE) and Poisson-Hamiltonian Integrators (PHI). Through experiments on some Poisson-Hamiltonian systems, we showed that PHNNs tested with PHI ensure better long-term stability and structure preservation, while Euler-trained models excel in short-term accuracy. A hybrid approach – Euler training and PHI testing – achieves the best balance of accuracy and stability.

The numerical results show that Poisson structures allow neural networks to learn generalized Hamiltonian dynamics, improving energy conservation when PHI is used for testing. Although implicit integrators (such as PHI) preserve structure, they introduce small errors that affect learning accuracy. Euler-trained models are more accurate in the short term, but can diverge in long-term integrations. The results highlight the potential of combining geometric numerical integrators with machine learning to model dynamic systems without explicit governing equations. Future research will focus on adaptive integrators that dynamically adjust step sizes to improve accuracy and stability. In addition, the exploration of Lie-Poisson neural networks [7] can provide further insight into geometric deep learning approaches for structure-preserving models. Finally, applications to control systems, plasma physics, and quantum mechanics represent promising directions for expanding the scope of PHNNs.

Acknowledgments. The authors were financially supported by the Fundação para a Ciência e a Tecnologia (FCT) under the scope of the project UID/00324 – Center for Mathematics of the University of Coimbra. Gonçalo Oliveira acknowledges FCT for support under the Ph.D. Scholarship UIDP/00324/2020. We also thank Eduardo Dias for his help in reviewing an earlier version.

References

1. A. Chattopadhyay, P. Hassanzadeh, and S. Pasha. Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data. *Scientific Reports*, 10, 01 2020.
2. Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou. Symplectic Recurrent Neural Networks. In *International Conference on Learning Representations*, 2020.
3. O. Cosserat. Symplectic groupoids for Poisson integrators. *Journal of Geometry and Physics*, 186:104751, April 2023.
4. O. Cosserat, C. Laurent-Gengoux, and V. Salnikov. Numerical Methods in Poisson Geometry and their Application to Mechanics. *Mathematics and Mechanics of Solids*, 29(5):904–923, 2024.

5. M. Crainic, R.L. Fernandes, and I. Mărcuţ. *Lectures on Poisson Geometry*. Graduate Studies in Mathematics. American Mathematical Society, 2021.
6. J. R. K. K. Dabbakuti and B. L. Gundapaneni. Application of Singular Spectrum Analysis Using Artificial Neural Networks in TEC Predictions for Ionospheric Space Weather. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(12):5101–5107, 2019.
7. C. Eldred, F. Gay-Balmaz, S. Huraka, and V. Putkaradze. Lie-Poisson neural networks (LPNets): data-based computing of Hamiltonian systems with symmetries. *Neural Netw.*, 173:20, 2024. Id/No 106162.
8. F. Gaya-Morey, J. M. Buades-Rubio, P. Palanque, R. Lacusta, and C. Manresa-Yee. Deep Learning-Based Facial Expression Recognition for the Elderly: A systematic review. *arXiv preprint arXiv:2502.02618*, 2025.
9. B. Gheflati and H. Rivaz. Vision Transformer for Classification of Breast Ultrasound Images. *arXiv preprint arXiv:2110.14731*, 2025.
10. M. Goswami, S. Dey, A. Mukherjee, S. Mohanty, and P.K. Pattnaik. Convolutional Neural Network Segmentation for Satellite Imagery Data to Identify Landforms Using U-Net Architecture. *arXiv preprint arXiv:2502.05476*, 2025.
11. S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. *arXiv preprint arXiv:1906.01563*, 2019.
12. E. Hairer, C. Lübich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2nd edition, 2006.
13. D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2017.
14. P. Libermann and C.M. Marle. *Symplectic Geometry and Analytical Mechanics*. Mathematics and Its Applications. Springer Netherlands, 1987.
15. J. Song, W. Cao, and W. Zhang. Fenn: Feature-enhanced neural network for solving partial differential equations involving fluid mechanics. *arXiv preprint arXiv:2501.11957*, 2025.