Direct Poisson neural networks: learning non-symplectic mechanical systems

OPEN ACCESS
IOP Publishing
Journal of Physics A: Mathematical and Theoretical
J. Phys. A: Math. Theor. **56** (2023) 495201 (25pp)
https://doi.org/10.1088/1751-8121/ad0803

Direct Poisson neural networks: learning non-symplectic mechanical systems

Martin Šípka[1], Michal Pavelka[1,*], Oğul Esen[2] and Miroslav Grmela[3]

[1] Mathematical Institute, Faculty of Mathematics and Physics, Charles University, Sokolovská 83, 18675 Prague, Czech Republic
[2] Department of Mathematics, Gebze Technical University, 41400 Gebze, Kocaeli, Turkey
[3] École Polytechnique de Montréal, C.P. 6079 suc. Centre-ville, Montréal H3C 3A7 Québec, Canada

E-mail: pavelka@karlin.mff.cuni.cz

Received 9 May 2023; revised 12 October 2023
Accepted for publication 30 October 2023
Published 15 November 2023

Abstract

In this paper, we present neural networks learning mechanical systems that are both symplectic (for instance particle mechanics) and non-symplectic (for instance rotating rigid body). Mechanical systems have Hamiltonian evolution, which consists of two building blocks: a Poisson bracket and an energy functional. We feed a set of snapshots of a Hamiltonian system to our neural network models which then find both the two building blocks. In particular, the models distinguish between symplectic systems (with non-degenerate Poisson brackets) and non-symplectic systems (degenerate brackets). In contrast with earlier works, our approach does not assume any further *a priori* information about the dynamics except its Hamiltonianity, and it returns Poisson brackets that satisfy Jacobi identity. Finally, the models indicate whether a system of equations is Hamiltonian or not.

Keywords: machine learning, Hamiltonian mechanics, non-symplectic, neural networks, Poisson

(Some figures may appear in colour only in the online journal)

* Author to whom any correspondence should be addressed.

1. Introduction

The estimation of unknown parameters in physics and engineering is a standard step in many well-established methods and workflows. One usually starts with a model—a set of assumptions and equations that are considered given and then, based on the available data, estimates the exact form of the evolution equations for the system of interest. As an example, we can consider a situation where we need to estimate the mass of a star far away based on its interaction with light [1], or when the moments of inertia of an asteroid are inferred from its rotations [2]. The assumptions can be of varying complexity and the method for parameter estimation should be therefore adequately chosen.

Techniques for machine learning of dynamical systems have sparked significant interest in recent years. With the rise of neural network-related advances, several methods have been developed for capturing the behavior of dynamical systems, each with its advantages and drawbacks. A symbolic approach (for instance [3]) allows us to learn precise symbolic form of equations from the predefined set of allowed operations. This can be often the most efficient approach that frequently leads to an exact match between the learned and target system, but the class of captured equations is by definition limited by the algebraic operations we consider as candidates.

Alternatively, one can learn directly the equations of motion

$$\dot{\mathbf{x}} = f(\mathbf{x}, \theta) \tag{1}$$

by learning f parameterized by weights θ. The function can be represented by any function approximator, in many cases by a neural network. Although this approach is very general, it does not incorporate any known physics into our procedure. There is no concept of energy of the system, no quantities are implicitly conserved, and the method thus might produce unphysical predictions. A remedy is the concept of physics-informed machine learning that constrains the neural network models so that they obey some required laws of physics [4, 5]. In particular, models of mechanical systems, which can be described by Hamiltonian mechanics, preserve several physical quantities like energy or angular momentum, as well as geometric quantities (for instance the symplectic two-form) that ensure the self-consistency of the systems. A neural-network model learning a Hamiltonian system from its trajectories that is compatible with the underlying geometry without any *a priori* knowledge about the system has been missing, to the best of our knowledge, and it is the main purpose of the current manuscript to introduce it. Moreover, we present several models that vary in how strictly they reproduce the underlying geometry and the degree to which these models learn a system can be used to estimate whether the system is Hamiltonian or not.

In [6], Cranmer *et al* demonstrate that for more complex, namely multiparticle systems, it is actually beneficial to merge symbolic and neural network parameterized fitting. The construction of a reduced representation subsequently fit by a symbolic expression yields interesting results in Hamiltonian mechanics and cosmology. This approach calls for tailored neural network system representations that could be easily learned and are accurate, representing the symmetries and constraints of the investigated system.

Geometrization of a dynamical system. A dynamical system is described by a differential equation, in particular, a mechanical system obeys Hamiltonian evolution equations. These equations are of geometric origin that is invariant with respect to changes of coordinates and which is preserved during the motion of the system. The geometry of Hamiltonian systems goes back to Lie and Poincaré [7, 8]. Modern approaches extend to infinite-dimensional

systems and provide foundations for many parts of nowadays physics [9–12]. Formulating a mechanical system geometrically typically means finding a bracket algebra (such as symplectic, Poisson, Jacobi, Leibniz, etc) and a generating function (such as Hamiltonian, energy, or entropy). The bracket is generally required to satisfy some algebraic conditions (Jacobi identity, Leibniz identity, etc). However, there is no general algorithmic way to obtain the Hamiltonian formulation (even if it exists) of a given system by means of analytical calculations. So such an analysis is proper for machine learning.

Apart from Hamiltonian mechanics, one can also include dissipation [13–15] or extend the learning of Hamiltonian systems to control problems [16]. Such an approach then, with the suitable choice of integrator ensures the conservation of physically important quantities, such as energy, momentum or angular momentum.

A reversible system is a candidate for being a Hamiltonian system. For a reversible system, the beginning point could be to search for a symplectic manifold and a Hamiltonian function. Learning the symplectic character (if it exists) of a physical system (including particles in potential fields, pendulums of various complexities) can be done utilizing neural networks, see, for example, [17, 18]. Another possibility is to learn the dynamics by minimizing the action, which leads to Lagrangian neural networks [19], and the dynamics is then symplectic as well. The symplectic geometry exists only in even-dimensional models and due to the non-degeneracy criteria, it is very rigid. A generalization of symplectic geometry is Poisson geometry where the non-degeneracy requirement is relaxed [20, 21]. In Poisson geometry, there exists a Poisson bracket (defining an algebra on the space of functions) satisfying the Leibniz and the Jacobi identities. This generalization permits (Hamiltonian) analysis in odd dimensions. The degenerate character of Poisson geometry brings some advantages for the investigation of (total, or even super) integrability of the dynamics. Poisson non-symplectic dynamics can be generated also by variation of action, but only by adding constraints, which is also a way to learn mechanics by neural networks [22].

The point of this paper is to better learn Hamiltonian mechanics. We build neural networks that encode the building blocks of Hamiltonian dynamics admitting Poisson geometry. According to the Darboux–Weinstein theorem [23] for a Poisson manifold, there are canonical coordinates which make the Poisson bivector (determining the bracket) constant. This formulation has also geometric implications, since it determines the symplectic foliation of the Poisson manifold (see more details in the main body of this paper). Recently, Poisson neural networks (abbreviated as PNNs) were proposed to learn Hamiltonian systems [24] by transforming the system in the Darboux–Weinstein coordinates. But, for many physical systems, the Poisson bracket is far from being in the canonical coordinates, and the dimension of the symplectic part of the Darboux–Weinstein Poisson bivector may be *a priori* unknown. For n-dimensional physical models, Jacobi identity, which ensures consistency of the underlying Poisson bivector, is a system of PDEs. To determine the Poisson structure, one needs to solve this system analytically, which is usually difficult, or enforce its validity while learning the Poisson bivector.

The goal of this work. The novel result of this paper is what we call a direct PNN (abbreviated as DPNN) which learns the Poisson structure without assuming any particular form of the Poisson structure such as Darboux–Weinstein coordinates. Instead, DPNN learns directly in the coordinate system in which the data are provided. There are several advantages of DPNN: **(i)** We do not need to know *a priori* the degeneracy level of the Poisson structure (or in other terms the dimensions of the symplectic foliations of the Poisson manifold) **(ii)** it is easier to learn the Hamiltonian (energy), and **(iii)** Jacobi identity is satisfied on the data, not only in a representation accessible only through another neural network (an invertible neural network

in [24]). DPNN learns Poisson systems by identifying directly the Poisson bivector and the Hamiltonian as functions of the state variables.

We actually provide three flavors of DPNNs. The least-informed flavor directly learns the Hamiltonian function and the Poisson bivector, assuming its skew-symmetry but not the Jacobi identity. Another flavor adds squares of Jacobi identity to the loss function and thus softly imposes its validity. The most geometry-informed flavor automatically satisfies Jacobi identity by building the Poisson bivector as a general solution to Jacobi identity in three dimensions (3D). While the most geometry-informed version is typically most successful in learning Hamiltonian systems, it is restricted to 3D systems, where the general solution of Jacobi identity is available. The second flavor is typically a bit less precise, and the least-informed flavor is usually the least precise, albeit still being able to learn Hamiltonian systems to a good degree of precision.

Interestingly, when we try to learn a non-Hamiltonian model by these three flavors of DPNNs, the order of precision is reversed and the least-informed flavor becomes most precise. The order of precision of the DPNNs flavors thus indicates whether a system of equations is Hamiltonian or not.

Section 2 recalls Poisson dynamics, in particular symplectic dynamics, rigid body (RB) mechanics, Shivamoggi equations, and evolution of heavy top (HT). Section 3 contains an introduction to neural networks to facilitate reading of this manuscript. It also introduces DPNNs and illustrates their use on learning Hamiltonian systems. Finally, section 4 shows DPNNs applied on a non-Hamiltonian system (dissipative RB).

2. Hamiltonian dynamics on Poisson geometry

2.1. General formulation

A Poisson bracket on a manifold $\mathcal{M}$ (physically corresponding to the state space, for instance position and momentum of the body) is a skew-symmetric bilinear algebra on the space $\mathcal{F}(\mathcal{M})$ of smooth functions on $\mathcal{M}$ given by

$$\{\bullet,\bullet\} : \mathcal{F}(\mathcal{M}) \times \mathcal{F}(\mathcal{M}) \to \mathcal{F}(\mathcal{M}). \tag{2}$$

Poisson brackets satisfy the Leibniz rule

$$\{F, HG\} = \{F, H\}\, G + H\, \{F, G\}, \tag{3}$$

and the Jacobi identity,

$$\{F, \{H, G\}\} + \{H, \{G, F\}\} + \{G, \{F, H\}\} = 0, \tag{4}$$

for arbitrary functions F, H and G, [12, 20, 21]. A manifold equipped with a Poisson bracket is called a Poisson manifold and is denoted by a two-tuple $(\mathcal{M}, \{\bullet,\bullet\})$. A function C is called a Casimir function if it commutes with all other functions that is $\{F, C\} = 0$ for all F. For instance, the magnitude of the angular momentum of an RB is a Casimir function.

Hamiltonian dynamics. Hamiltonian dynamics can be seen as evolution on a Poisson manifold. For a Hamiltonian function (physically energy) H on $\mathcal{M}$, the Hamiltonian vector field and Hamilton's equation are

$$X_H(F) := \{F, H\}, \qquad \dot{\mathbf{x}} = X_H(\mathbf{x}) = \{\mathbf{x}, H\}, \tag{5}$$

respectively, where $\mathbf{x} \in \mathcal{M}$ is a parametrization of manifold $\mathcal{M}$. The algebraic properties of the Poisson bracket have some physical consequences. Skew-symmetry implies energy conservation,

$$\dot{H} = \{H, H\} = 0, \tag{6}$$

while the Leibniz rule ensures that the dynamics does not depend on biasing the energy by a constant. Referring to a Poisson bracket, one may determine the Poisson bivector field according to

$$L(\mathrm{d}F, \mathrm{d}H) := \{F, H\}. \tag{7}$$

This identification makes it possible to define a Poisson manifold by a tuple $(\mathcal{M}, L)$ consisting of a manifold and a Poisson bivector. In this notation, the Jacobi identity can be rewritten as $\mathcal{L}_{\mathbf{X}_H} L = 0$, that is the Lie derivative of the Poisson bivector with respect to the Hamiltonian vector field is zero [25]. In other words, the Jacobi identity expresses the self-consistency of the Hamiltonian dynamics in the sense that both the building blocks (Hamiltonian function and the bivector field) are constant along the evolution.

Assuming a local coordinate system $\mathbf{x} = (x^i)$ on $\mathcal{M}$, Poisson bivector determines Poisson matrix $L = [L^{kl}]$ which enables us to write [26]

$$L = L^{kl} \frac{\partial}{\partial x^k} \wedge \frac{\partial}{\partial x^l}. \tag{8}$$

In this realization, the Poisson bracket and Hamilton's equations are written as

$$\{F, H\} = L^{kl} \frac{\partial F}{\partial x^k} \frac{\partial H}{\partial x^l}, \qquad \dot{x}^k = L^{kl} \frac{\partial H}{\partial x^l}, \tag{9}$$

respectively. Here, we have assumed summation over the repeated indices. Further, the Jacobi identity (4) turns out to be the following system of partial differential equations

$$J^{ijl} = L^{kl} \frac{\partial L^{ij}}{\partial x^k} + L^{ki} \frac{\partial L^{jl}}{\partial x^k} + L^{kj} \frac{\partial L^{li}}{\partial x^k} = 0. \tag{10}$$

The left-hand side of this equation is called Jacobiator, and in the case of Hamiltonian systems, it is equal to zero. Jacobi identity (10) is a system of differential equations consisting of PDEs. In 3D systems (for instance the RB, that has three degrees of freedom), Jacobi identity (10) is a single PDE whose general solution is known [27]. In four-dimensional (4D) systems (for instance a particle moving in a plane, or the Shivamoggi equations), Jacobi identity (10) consists of four PDEs, and there are some partial results, but for an arbitrary n, according to our knowledge, there is no general solution yet. We shall focus on 3D, 4D, and six-dimensional (6D) (HT or particle in a volume) cases in the upcoming subsections.

Symplectic manifolds. If there is no non-constant Casimir function for a Poisson manifold, then it is also a symplectic manifold. Although we can see symplectic manifolds as examples of Poisson manifolds, it is possible to define a symplectic manifold in a direct way without referring to a Poisson manifold. A manifold $\mathcal{M}$ is called symplectic if it is equipped with a closed non-degenerate two-form (called a symplectic two-form) Ω. A two-form is called non-degenerate when

$$\Omega(X, Y) = 0, \qquad \forall X \in \mathfrak{X}(\mathcal{M}) \tag{11}$$

implies $Y = 0$. A two-form is closed when being in the kernel of de Rham exterior derivative, $\mathrm{d}\Omega = 0$. A Hamiltonian vector field X_H on a symplectic manifold $(\mathcal{M}, \Omega)$ is defined as

$$\iota_{X_H}\Omega = \mathrm{d}H, \tag{12}$$

where ι is the contraction operator (more precisely the interior derivative) [25]. Referring to a symplectic manifold one can define a Poisson bracket

$$\{F, H\} := \Omega\left(X_F, X_H\right), \tag{13}$$

where the Hamiltonian vector fields are defined through equation (12). The closedness of the symplectic two-form Ω guarantees the Jacobi identity (4). The non-degeneracy condition of Ω puts an extra condition to the bracket (13) that the Casimir functions are only the constant functions, in contrast with Poisson manifolds, which may have also non-constant Casimirs. The Darboux–Weinstein coordinates show more explicitly the relationship between Poisson and symplectic manifolds in a local picture.

Darboux–Weinstein coordinates. We start with $n = (2m + k)$-dimensional Poisson manifold $\mathcal{M}$ equipped with Poisson bivector L. Near every point of the Poisson manifold, the Darboux–Weinstein coordinates $(x^i) = (q^a, p_b, u^\alpha)$ (here a runs from 1 to m, and α runs from 1 to k) give a local form of the Poisson bivector

$$L = \frac{\partial}{\partial q^a} \wedge \frac{\partial}{\partial p_a} + \frac{1}{2}\lambda^{\alpha\beta}\frac{\partial}{\partial u^\alpha} \wedge \frac{\partial}{\partial u^\beta} \tag{14}$$

with the coefficient functions $\lambda^{\alpha\beta}$ equal zero at the origin. If $k = 0$ in the local formula (14), then there remains only the first term on the right-hand side and the Poisson manifold turns out to be a $2m$-dimensional symplectic manifold. Newtonian mechanics, for instance, fits this kind of realization. On the other hand, if $m = 0$ in (14), then there remains only the second term which is a full-degenerate Poisson bivector. A large class of Poisson manifolds is of this form, namely Lie–Poisson structure on the dual of a Lie algebra including RB dynamics, Vlasov dynamics, etc. In general, Poisson bivectors have both the symplectic part as well as the fully degenerate part, for instance, the HT dynamics in section 2.4.

When the Poisson bivector is non-degenerate, it generates a symplectic Poisson bracket, and it commutes with the canonical Poisson bivector

$$\mathbf{L}_{\mathrm{can}} = \begin{pmatrix} 0 & \mathbf{I} \\ -\mathbf{I} & 0 \end{pmatrix} \tag{15}$$

in the sense that

$$\mathbf{L} \cdot \mathbf{L}_{\mathrm{can}} - \mathbf{L}_{\mathrm{can}} \cdot \mathbf{L} = 0. \tag{16}$$

This compatibility condition is employed later in section 3 to measure the error of DPNNs when learning symplectic Poisson bivectors.

2.2. 3D Hamiltonian dynamics

In this subsection, we focus on 3D Poisson manifolds, following [28–30]. One of the important observations in 3D is the isomorphism between the space of vectors and the space of skew-symmetric matrices given by

$$\mathbf{L} = \begin{pmatrix} 0 & -J_z & J_y \\ J_z & 0 & -J_x \\ -J_y & J_x & 0 \end{pmatrix} \leftrightarrow \mathbf{J} = (J_x, J_y, J_z)\,. \tag{17}$$

This isomorphism lets us write Jacobi identity (10) as a single scalar equation

$$\mathbf{J} \cdot (\nabla \times \mathbf{J}) = 0, \tag{18}$$

see, for example, [29, 31–33]. The general solution of Jacobi identity (18) is

$$\mathbf{J} = \frac{1}{\phi} \nabla C \tag{19}$$

for arbitrary functions ϕ and C, where C is a Casimir function. Hamilton's equation then takes the particular form

$$\dot{\mathbf{x}} = \mathbf{J} \times \nabla H = \frac{1}{\phi} \nabla C \times \nabla H. \tag{20}$$

Note that by changing the roles of the Hamiltonian function H and the Casimir C one can arrive at another Hamiltonian structure for the same system. In this case, the Poisson vector is defined as $\mathbf{J} = -(1/\phi)\nabla H$ and the Hamiltonian function is C. This is an example of a bi-Hamiltonian system, that manifests integrability [27, 34–36]. A bi-Hamiltonian system admits two different but compatible Hamilton formulations. In 3D, two Poisson vectors, say $\mathbf{J}_1$ and $\mathbf{J}_2$ are compatible if

$$\mathbf{J}_1 \cdot (\nabla \times \mathbf{J}_2) = \mathbf{J}_2 \cdot (\nabla \times \mathbf{J}_1)\,. \tag{21}$$

Indeed, if a 3D Hamiltonian system admits two Poisson bivectors (represented by two vector $\mathbf{J}_1$ and $\mathbf{J}_2$) and two Hamiltonians H_1 and H_2,

$$\dot{\mathbf{x}} = \mathbf{J}_1 \times \nabla H_1 = \mathbf{J}_2 \times \nabla H_2, \tag{22}$$

then both the Hamiltonians are conserved and thus H_2 serves as a Casimir function in the formulation with $\mathbf{J}_1$ while H_1 being a Casimir for $\mathbf{J}_2$. Compatibility condition (21) then follows from the solution of Jacobi identity (19). This compatibility condition will be used later in section 3 to measure the error of learning Poisson bivectors in 3D by DPNNs.

RB dynamics. Let us consider an example of a 3D Hamiltonian system, a freely rotating RB. The state variable $\mathbf{M} \in \mathcal{M}$ is the angular momentum in the frame of reference co-rotating with the RB. The Poisson structure is

$$\{F, H\}^{(\mathrm{RB})}(\mathbf{M}) = -\mathbf{M} \cdot \frac{\partial F}{\partial \mathbf{M}} \times \frac{\partial H}{\partial \mathbf{M}}, \tag{23}$$

see [37]. Poisson bracket (23) is degenerate because it preserves any function of the magnitude of **M**. The Hamiltonian function is the energy

$$H = \frac{1}{2}\left(\frac{M_x^2}{I_x} + \frac{M_y^2}{I_y} + \frac{M_z^2}{I_z}\right), \tag{24}$$

where I_x, I_y and I_z are moments of inertia of the body. In this case, Hamilton's equation

$$\dot{\mathbf{M}} = \mathbf{M} \times \frac{\partial H}{\partial \mathbf{M}} \tag{25}$$

gives Euler's RB equation [38].

2.3. 4D Hamiltonian dynamics

In 4D, we consider the following local coordinates $(u,x,y,z) = (u,\mathbf{x})$. A skew-symmetric matrix L can be identified with a couple of vectors $\mathbf{U} = (U^1,U^2,U^3)$ and $\mathbf{V} = (V^1,V^2,V^3)$ as

$$L = \begin{pmatrix} 0 & -U^1 & -U^2 & -U^3 \\ U^1 & 0 & -V^3 & V^2 \\ U^2 & V^3 & 0 & -V^1 \\ U^3 & -V^2 & V^1 & 0 \end{pmatrix}. \tag{26}$$

After this identification, Jacobi identity (10) turns out to be a system of PDEs consisting of four equations [39]

$$\partial_u(\mathbf{U}\cdot\mathbf{V}) = \mathbf{V}\cdot(\partial_u\mathbf{U} - \nabla\times\mathbf{V}), \tag{27a}$$
$$\nabla(\mathbf{U}\cdot\mathbf{V}) = \mathbf{V}(\nabla\cdot\mathbf{U}) - \mathbf{U}\times(\partial_u\mathbf{U} - \nabla\times\mathbf{V}). \tag{27b}$$

Note that L is degenerate (its determinant being zero) if and only if $\mathbf{U}\cdot\mathbf{V} = 0$. So, for degenerate Poisson matrices, the Jacobi identity is satisfied if

$$\nabla\cdot\mathbf{U} = 0, \qquad \partial_u\mathbf{U} - \nabla\times\mathbf{V} = 0. \tag{28}$$

Superintegrability. Assume that a given dynamics admits two time-independent first integrals, say H_1 and H_2. Then, when vectors **U** and **V** have the form

$$\mathbf{U} = \nabla H_1 \times \nabla H_2, \qquad \mathbf{V} = \partial_u H_1 \nabla H_2 - \partial_u H_2 \nabla H_1, \tag{29}$$

they constitute a Poisson bivector, and in particular Jacobi identity (28) is satisfied. Functions H_1 and H_2 are Casimir functions. For a Hamiltonian function H_3, the Hamiltonian dynamics is

$$\dot{u} = -(\nabla H_1 \times \nabla H_2)\cdot\nabla H_3, \tag{30a}$$
$$\mathbf{x} = (\nabla H_1 \times \nabla H_2)\,\partial_u H_3 + (\nabla H_2 \times \nabla H_3)\,\partial_u H_1 + (\nabla H_3 \times \nabla H_1)\,\partial_u H_2. \tag{30b}$$

By permuting the roles of the functions H_1, H_2, and H_3 (two of them are Casimirs and one of them is Hamiltonian), one arrives at two additional Poisson realizations of the dynamics. This is the case of a superintegrable (tri-Hamiltonian) formulation, [40].

Two Poisson bivectors of a superintegrable 4D Hamiltonian system must satisfy the compatibility condition

$$\mathbf{U}_1 \cdot \mathbf{V}_2 + \mathbf{V}_1 \cdot \mathbf{U}_2 = 0 \tag{31}$$

where $\mathbf{U}_{1,2}$ and $\mathbf{V}_{1,2}$ are the vectors identified from formula (26). We shall use this compatibility condition to measure the error of DPNNs when learning Poisson bivectors in 4D.

Shivamoggi equations. An example of a 4D Poisson (and non-symplectic) system is the Shivamoggi equations, which arise in the context of magnetohydrodynamics,

$$\dot{u} = -uy, \qquad \dot{x} = zy, \qquad \dot{y} = zx - u^2, \qquad \dot{z} = xy, \tag{32}$$

see [41, 42]. The first integrals of this system of equations are

$$H_1 = x^2 - z^2, \qquad H_2 = z^2 + u^2 - y^2, \qquad H_3 = u\,(z+x)\,. \tag{33}$$

Vectors $\mathbf{U}_i$ and $\mathbf{V}_i$ of Poisson matrices $N^{(i)}$ $(i = 1,2,3)$ for the Hamiltonian functions H_1, H_2, and H_3 are

$$\begin{aligned}
&\mathbf{U}_1 = 2u\,(-y,z,y)\,, \qquad &&\mathbf{V}_1 = 2\left(u^2, y\,(x+z)\,, u^2 - z\,(x+z)\right),\\
&\mathbf{U}_2 = 2\,(x+z)\,(0,u,0)\,, \qquad &&\mathbf{V}_2 = 2\,(x+z)\,(x,0,-z)\,,\\
&\mathbf{U}_3 = -4\,(yz,zx,xy)\,, \qquad &&\mathbf{V}_3 = 4u\,(-x,0,z)\,,
\end{aligned} \tag{34}$$

respectively. Note that all these three Poisson matrices are degenerate, since $\mathbf{U}_i \cdot \mathbf{V}_i = 0$ holds for all $i = 1,2,3$. The equations of motion can be written as

$$X = \vartheta N^{(1)} \bar{\nabla} H_1 = \vartheta N^{(2)} \bar{\nabla} H_2 = \vartheta N^{(3)} \bar{\nabla} H_3, \quad \vartheta = -\frac{1}{4\,(x+z)}$$

up to multiplication with a conformal factor ϑ in all three cases. Note that the 4D gradient is denoted by $\bar{\nabla} H = (\partial_u H, \partial_x H, \partial_y H, \partial_z H)$.

2.4. Semi-direct extension to a 6D system

6D Hamiltonian systems can again be symplectic or non-symplectic (degenerate). The former case is represented by a particle in 3D (P3D) while the latter is for instance the HT dynamics. Since the evolution of a P3D is canonical and thus analogical to the 2D dynamics, we shall recall only the HT dynamics.

A supported rotating RB in a uniform gravitational field is called HT [43]. The mechanical state of the body is described by the position of the center of mass $\mathbf{r}$ and angular momentum $\mathbf{M}$. In this case, the Poisson bracket is

$$\{F,G\}^{(\mathrm{HT})}(\mathbf{r},\mathbf{M}) = -\mathbf{M} \cdot \left(\frac{\partial F}{\partial \mathbf{M}} \times \frac{\partial G}{\partial \mathbf{M}}\right) - \mathbf{r} \cdot \left(\frac{\partial F}{\partial \mathbf{M}} \times \frac{\partial G}{\partial \mathbf{r}} - \frac{\partial G}{\partial \mathbf{M}} \times \frac{\partial F}{\partial \mathbf{r}}\right). \tag{35}$$

Even though the model is even dimensional, it is not symplectic. Two non-constant Casimir functions are $\mathbf{r}^2$ and $\mathbf{M} \cdot \mathbf{r}$. In this case, we assume the Hamiltonian function as

$$H = \frac{1}{2}\left(\frac{M_x^2}{I_x} + \frac{M_y^2}{I_y} + \frac{M_z^2}{I_z}\right) + Mgl\mathbf{r} \cdot \boldsymbol{\chi}, \tag{36}$$

where $-g\boldsymbol{\chi}$ is the vector of gravitational acceleration. Hamilton's equation is then

$$\dot{\mathbf{M}} = \mathbf{M} \times \frac{\partial H}{\partial \mathbf{M}} + \mathbf{r} \times \frac{\partial H}{\partial \mathbf{r}} \tag{37a}$$

$$\dot{\mathbf{r}} = -\mathbf{r} \times \frac{\partial H}{\partial \mathbf{M}}. \tag{37b}$$

In the following sections, we apply DPNNs to the models here recalled and show that DPNNs are capable to extract the Poisson bivector and Hamiltonian from simulated trajectories of the models.

3. Learning Hamiltonian systems

Before going to the actual model recognition by DPNNs, let us briefly recall how neural networks work.

3.1. Introduction to neural networks

Neural networks serve as robust interpolators to high-dimensional functions. As they consist of several interconnected layers of neurons, they can be seen as compositions of high-dimensional mappings, which provides the robustness of approximation. Actually, the universal approximation theorem states roughly speaking that every continuous function from $\mathbb{R}^n$ to $\mathbb{R}^m$ can be approximated by a neural network with one hidden layer and a non-linear activation function [44–46]. These multi-layered architectures have shown a great capability for tasks like image recognition, natural language processing, and other applications of deep learning [47].

A prototype of such neural networks is the fully connected (or dense) network, see figure 1, where every neuron in a given layer is connected to every neuron in the subsequent layer. The mathematical operation each neuron typically carries out is an affine transformation of the inputs followed by the application of a non-linear activation function [48]. Parameters of the affine transformation (weights and biases) are then subject to the learning procedure, where they are sought so that a prescribed loss function is minimized.

The affine transformation between two consecutive layers is associated with weights (the strengths of particular neuron–neuron connections) and biases, which are added to the weighted sum of the inputs, turning a zero input into a non-zero value. The activation function plays a crucial role because it provides non-linearity to the neural network, enabling it to approximate more complicated functions. In our approach, we use the softplus function,

$$\sigma_{\text{softplus}}(x) = \log\left(1 + e^{x}\right), \tag{38}$$

which ensures a differentiability of the outputs [49, 50].

Training a neural network then involves tuning its weights and biases to minimize the difference between the actual and the expected outputs. This difference is quantified by a so called loss function. The weights and biases are then adapted in such a way that the loss function is minimized by a backpropagation algorithm, which computes the gradient of the loss function with respect to the parameters by applying the chain rule of calculus [51] and then shifts the parameters along the gradient. The simplest implementation called stochastic gradient descent (SGD) works by splitting a large dataset into smaller chunks—minibatches and then performing incremental updates along the loss gradient of the corresponding batch. Many variations of this algorithm exist such as the Adam algorithm (short for adaptive moment estimation)

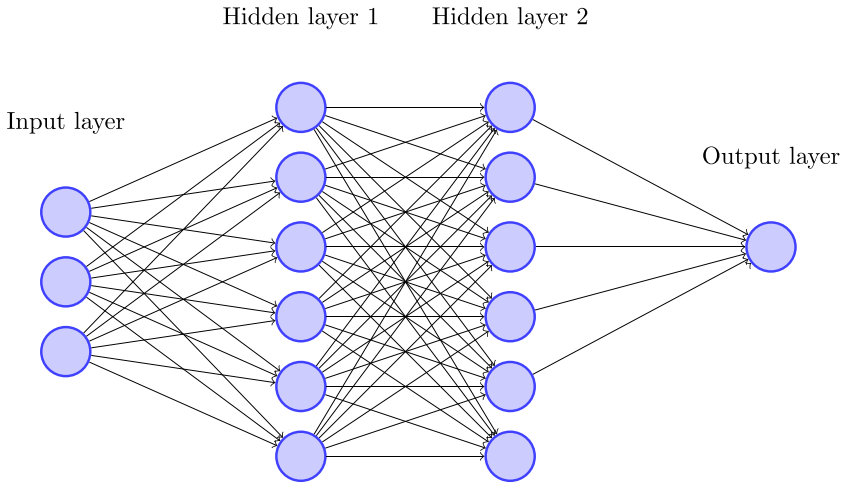


Figure 1. The structure of the energy neural network used in this paper. When the dimension of the learned data is three (as for instance in rigid body mechanics), the input layer has three neurons. The output layer is just one number (the value of energy). Inbetween the input and output layers, hidden layers provide the necessary complexity so that the combined mappings from one layer to another is rich enough to approximate the energy. The structure of the neural network for the $\mathbf{L}$ bivector is the same except for the size of the output layer which is given by the number of independent components of that bivector.

[52] that works like SGD but utilizes momentum term to make the optimization smoother. We shall use Adam optimizer in this work. Although much time could be spent on the various techniques of deep learning, we refer an interested reader to book [48], and proceed to the actual application in Hamiltonian systems.

3.2. Hamiltonian model recognition by machine learning

When we have a collection of snapshots of a trajectory of a Hamiltonian system, how to identify the underlying mechanics? In other words, how to recognize the model (Poisson bivector and energy) from the snapshots? Machine learning provides a robust method for such task. Machine learning has previously been shown to reconstruct GENERIC models [13, 14, 53], but the Poisson bivector is typically known and symplectic. PNNs [24] provide a method for learning also non-symplectic mechanics, which however relies on the identification of dimension of the symplectic subdynamics in the Darboux–Weinstein coordinates and on a transformation to the coordinates. Here, we show a robust method that does not need to know the dimension of the symplectic subsystem and that satisfies Jacobi identity also in the coordinates in which the data are prescribed. Therefore, we refer to the method as DPNNs.

DPNNs learn Hamiltonian mechanics directly by training a model for the $\mathbf{L}(\mathbf{x})$ matrix and a model for the Hamiltonian $H(\mathbf{x})$ simultaneously. The neural network that encodes $\mathbf{L}(\mathbf{x})$ only learns the upper triangular part of $\mathbf{L}$ and skew-symmetry is then automatically satisfied. The network has one hidden fully connected layer (with 64 neurons) equipped with the softplus activation. The network that learns $H(\mathbf{x})$ has the same structure. The input layer has the dimension of the physical system (for instance three nodes in the case of RB), and dimension of the output layer is the number of independent components of the trained quantity (number of

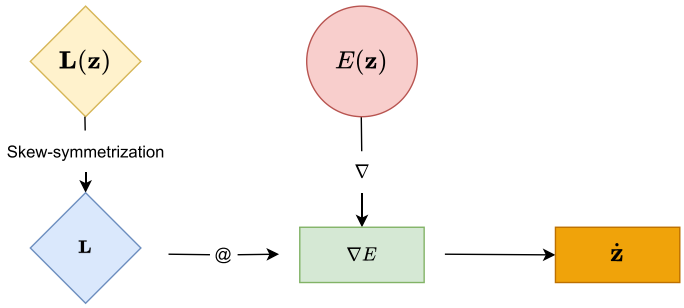


Figure 2. Scheme SJ (soft Jacobi) of the methods that learn both the energy and Poisson bivector.

independent components of the $\mathbf{L}$ bivector, or one when learning H). The actual learning was implemented within the standard framework PyTorch [54], using the Adam optimizer [52], and is publically available on Github [55].

The loss function contains squares of deviation of the training data and predicted trajectories that are obtained by the implicit midpoint rule (IMR) numerically solving the exact equations (for the training data) or Hamilton's equation with the trained models for $\mathbf{L}(\mathbf{x})$ and $H(\mathbf{x})$ (for the predicted data). Although such a model leads to a good match between the grand truth trajectories and predicted trajectories, it does not need to satisfy Jacobi identity. Therefore, we use also an alternative model where squares of the Jacobiator (10) are added to the loss function, which enforces Jacobi identity in a soft way, see figure 2. Finally, in 3D we know the form of the Poisson bivector since we have the general solution of Jacobi identity (20). In such a case, the neural network encoding $\mathbf{L}$ can be simplified to a network learning $C(\mathbf{x})$ and Jacobi identity is automatically satisfied, see figure 3. In summary, we use three methods:

- **(WJ)** Training $\mathbf{L}(\mathbf{x})$ and $H(\mathbf{x})$ *without* the Jacobi identity. The loss function consists only of the L_2 error measuring the discrepancy between the training steps and steps obtained by IMR iterations with bivector $\mathbf{L}$ and Hamiltonian $H(\mathbf{x})$ encoded by the networks:

$$\mathcal{L}_{\mathrm{WJ}} = c_{\mathrm{mov}} \sum_n | \left((\mathbf{x}_{n+1})_{\mathrm{exact}} - \mathbf{x}_n \right) - \left((\mathbf{x}_{n+1})_{\mathrm{NN}} - \mathbf{x}_n \right) |^2 \tag{39a}$$

  where n goes over all training steps. Each point $(\mathbf{x}_{n+1})_{\mathrm{NN}}$ is calculated from $\mathbf{x}_n$ by the IMR method (using the $\mathbf{L}$ and H that are being trained and encoded by the neural networks). Prefactor c_{mov} is used to scale the loss function to numerically advantageous values (typically we use $c_{\mathrm{mov}} = 10$).
- **(SJ)** Training $\mathbf{L}(\mathbf{x})$ and $H(\mathbf{x})$ with *soft* Jacobi identity, where the L_2-norm of the Jacobiator (10) is a part of the loss function. The loss function is thus

$$\mathcal{L}_{\mathrm{SJ}} = \mathcal{L}_{\mathrm{WJ}} + c_{\mathrm{Jac}} \sum_n \sum_{ijk} |J^{ijk}(\mathbf{x}_n)|^2, \tag{39b}$$

  where i, j, and k go over the dimension of the system (here 3, 4, or 6). Prefactor c_{Jac} is used to scale the loss function into better numericalvalues (typically we use $c_{\mathrm{Jac}} = 10$).

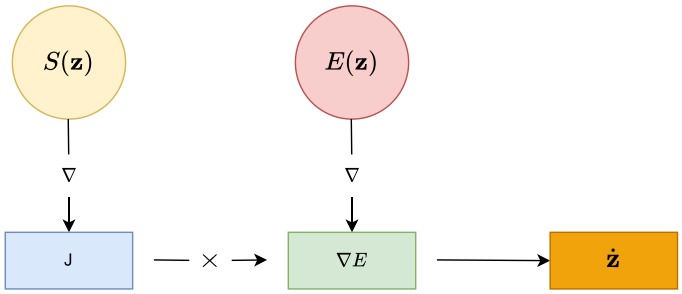


Figure 3. Scheme IJ (implicit Jacobi) of the learning method implicitly enforcing Jacobi identity.

- **(IJ)** Training $C(\mathbf{x})$ and $H(\mathbf{x})$ with *implicitly* valid Jacobi identity, based on the general solution of Jacobi identity in 3D (20). The loss function is the same as in the case of the WJ method, but this time Jacobi identity is valid automatically,

$$\mathcal{L}_{\mathrm{IJ}} = \mathcal{L}_{\mathrm{WJ}}. \tag{39c}$$

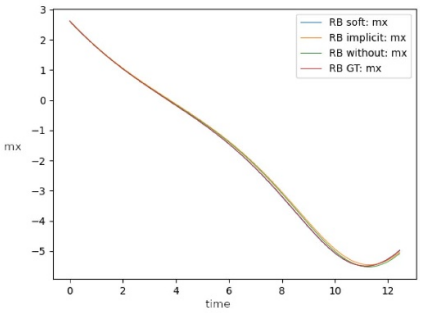
The training itself then proceeds in the following steps:

1. Simulation of the training and validation data. For a randomly generated set of initial conditions, we simulate a set of trajectories by IMR. These trajectories are then split into steps (couples of two consecutive states) and the collection of steps is split into a training set and a validation set (40% used for the validation set). The collection of steps is shuffled and split to batches (20 steps per a batch).
2. Parameters of the neural networks WJ, SJ, and IJ are trained by back-propagation on the training data (using the Adam optimizer), minimizing the loss function. The loss function differs in the three flavors of DPNNs, as in equation (39). Then, the loss function is evaluated on the validation data to report the errors. The learning rate was set to 0.001 while the number of epochs is chosen appropriately for each physical system.
3. A new set of initial conditions is randomly chosen and new trajectories are generated using IMR, which gives the ground truth (GT).
4. Trajectories with the GT initial conditions are simulated using the trained models for $\mathbf{L}$ and H and compared with GT.

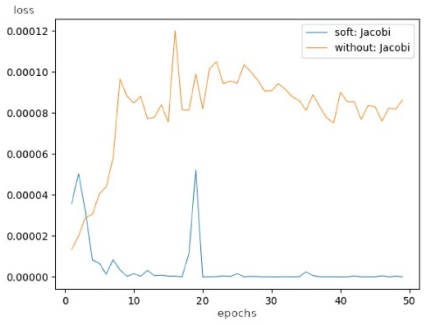
In the following sections, we illustrate this procedure for learning RB mechanics, a particle in two dimensions (P2D), Shivamoggi equations, a P3D, and HT dynamics.

3.3. Rigid body (RB)

Figure 4(a) shows a sample trajectory of RB dynamics (25) from the GT set, as well as trajectories with the same initial conditions, generated the DPNNs. The training was carried out on



(a) Comparison of an exact trajectory (GT) and trajectories obtained by integrating the learned models. All three methods fit the trajectories well. Here, as well in the subsequent figures, mx stands for the x-component of vector $\mathbf{m}$.



(b) Error of Jacobi identity on the validation set. The error of IJ is zero by construction, the error of SJ goes to zero while the error of WJ does not.

Figure 4. Rigid body: comparison of learned models (WJ, SJ, and IJ) with GT.

200 trajectories while GT consisted of 400 trajectories. Number of training epochs was 50. The initial conditions for the training and validation datasets were sampled uniformly from a ball of radius 11.18 (in the $\mathbf{m}$ variable), each trajectory consisted of 200 steps, and the moments of inertia were chosen as $I_x = 10.0$, $I_y = 20.0$, and $I_z = 40.0$.

Errors of the three learning methods (WJ, SJ, and IJ) are shown in table 1. All three methods were capable to learn the dynamics well. Figure 4(b) shows the norm of the Jacobiator evaluated on the validation set. Jacobiator is zero for IJ and small in SJ, while it does not go to zero in WJ. Therefore, IJ and SJ satisfy Jacobi identity while WJ does not.

Figure 5 shows the compatibility error of learning the Poisson bivector (21). All three methods learn the Poisson bivector well, but IJ is the most precise, followed by SJ and WJ. Finally, figure 6 shows errors in learning the trajectories $\mathbf{M}(t)$. All three methods learn the trajectories well, but in this case, the SJ method works slightly better.

Table 1 shows the medians of the errors for all the models and applied learning methods. N/A indicates quantities that are not to be conserved. Error $\Delta\mathbf{M}$ is calculated as the median of square deviation of $\mathbf{M}^2$ over all time steps. Error $\Delta\mathbf{r}$, $\Delta\mathbf{M}\cdot\mathbf{r}$, $\Delta\mathbf{M}^2$, and $\Delta\mathbf{r}^2$ are calculated analogically. Error $\Delta\mathbf{L}$ in the RB case is calculated as $\log_{10}$ of the L^2 norm of the compatibility condition (21), calculated for the learned $\mathbf{J}$ divided by its trace and multiplied by factor 1000 (using the exact $\mathbf{J}$ when generating the GT). In the P2D and P3D cases, where the Poisson bivector is symplectic, the error is calculated as $\log_{10}$ of squares of the symplecticity condition (16). In the case of Shivamoggi equations, the $\Delta\mathbf{L}$ error is the $\log_{10}$ of the squared superintegrable compatibility condition (31). $\Delta\det\mathbf{L}$ errors are medians of squares of learned $\det\mathbf{L}$, and in the Shivamoggi and the HT cases, the values are logarithmic, since determinants are supposed to be zero in those cases.

3.4. Particle in 2D (P2D)

A particle moving in a 2D potential field represents a 4D symplectic system. Initial conditions were sampled uniformly from balls of radius 11.18 in the momentum and 10.48 in the positions (200 points), and each trajectory had 200 steps. The Hamiltonian was chosen as $H(\mathbf{r},\mathbf{m}) = \mathbf{m}^2 + \mathbf{r}^2$. GT was generated in the same way, but with 400 sampling points.

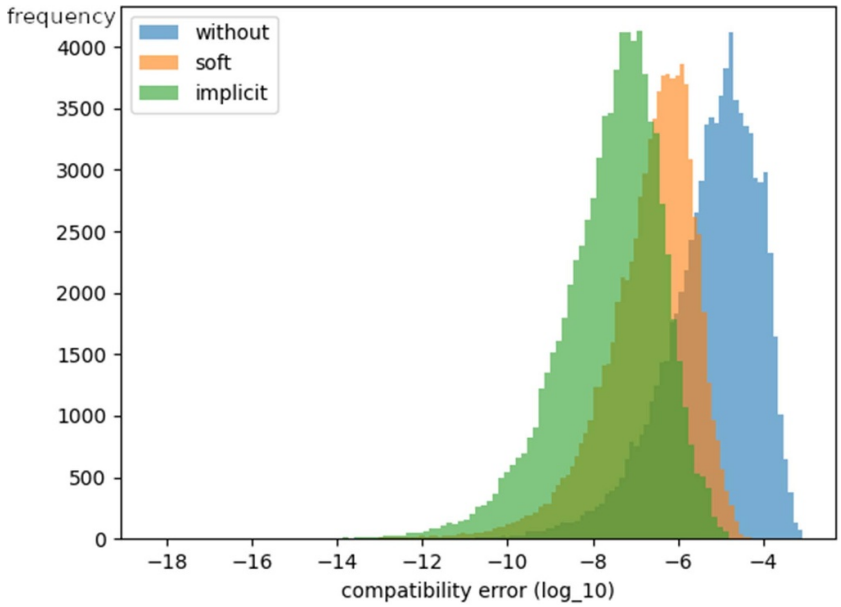


Figure 5. Rigid body: compatibility errors for RB evaluated as $\log_{10}$ of squares of the difference between the left hand side and right hand side of equality (21). That equality is satisfied if and only if the two Poisson bivectors describe the same Hamiltonian system. The distribution of errors is approximately log-normal. The compatibility error of the IJ method is the lowest, followed by SJ and WJ.

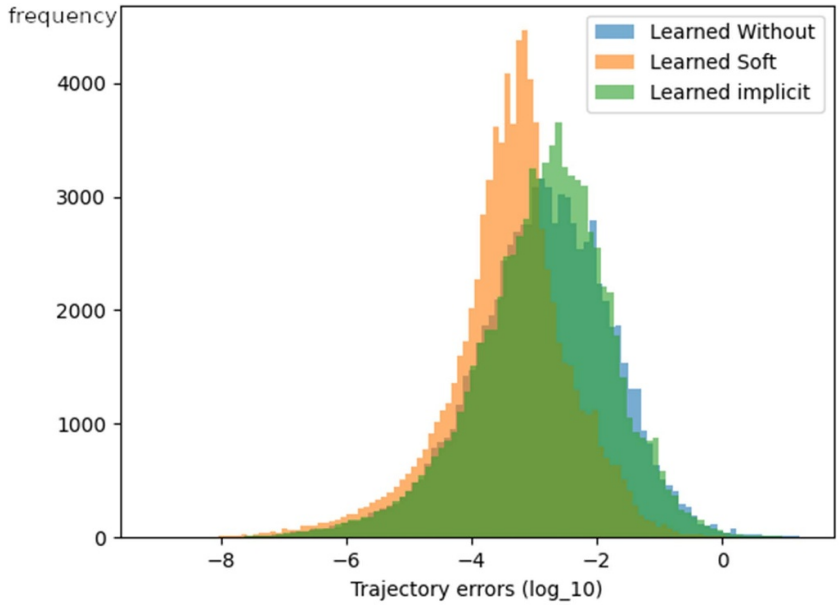


Figure 6. Rigid body: distribution of $\log_{10}$ of squares of errors in $\mathbf{M}$.

Table 1. Summary of the learning errors for a rigid body (RB), particle in two dimensions (P2D), Shivamoggi equations (Sh), particle in three dimensions (P3D), and heavy top (HT).

Mod.	Met.	$\Delta\mathbf{M}$	$\Delta\mathbf{r}$	$\Delta\mathbf{L}$	$\Delta\mathbf{M}^2$	$\Delta(\mathbf{r}\times\mathbf{M})$	$\Delta\mathbf{r}^2$	$\Delta(\mathbf{M}\cdot\mathbf{r})$	$\det\mathbf{L}$
RB	WJ	1.5×10^{-03}	N/A	-5.0	5.2×10^{-04}	N/A	N/A	N/A	0
	SJ	4.6×10^{-04}	N/A	-6.5	3.9×10^{-04}	N/A	N/A	N/A	0
	IJ	1.7×10^{-03}	N/A	-7.4	4.8×10^{-04}	N/A	N/A	N/A	0
P2D	WJ	3.2×10^{-03}	3.2×10^{-03}	-1.8	N/A	9.7×10^{-03}	N/A	N/A	0.83
	SJ	2.9×10^{-03}	2.9×10^{-03}	-1.8	N/A	8.6×10^{-03}	N/A	N/A	0.93
Sh	WJ	N/A	3.4×10^{-04}	-2.3	N/A	N/A	N/A	N/A	-2.1
	SJ	N/A	3.0×10^{-04}	-3.3	N/A	N/A	N/A	N/A	-3.4
P3D	WJ	1.9×10^{-02}	1.9×10^{-02}	-1.6	N/A	3.7×10^{-02}	N/A	N/A	0.91
	SJ	1.0×10^{-02}	9.1×10^{-03}	-1.9	N/A	3.5×10^{-02}	N/A	N/A	0.70
HT	WJ	3.1×10^{-02}	9.9×10^{-03}	N/A	N/A	N/A	1.9×10^{-03}	2.8×10^{-03}	-2.5
	SJ	1.6×10^{-02}	4.6×10^{-03}	N/A	N/A	N/A	1.3×10^{-03}	1.9×10^{-03}	-2.6

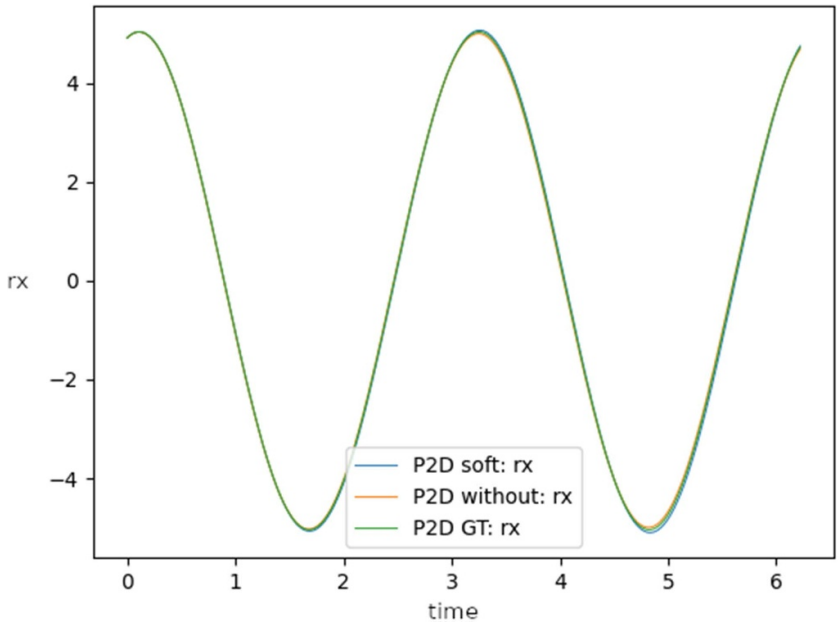


Figure 7. P2D: a sample trajectory. Both SJ and WJ learn the dynamics of a particle in two dimensions well. Here, *rx* is the *x*-component of the **r** vector.

The simulated trajectories were learned by WJ and SJ methods (each with 50 epochs). No implicit IJ method was used because no general solution of Jacobi identity in 4D is available that would work for both the degenerate and symplectic Poisson bivectors. Results of the learning are in table 1, and both WJ and SJ learn the dynamics comparably well. Figure 7 shows a

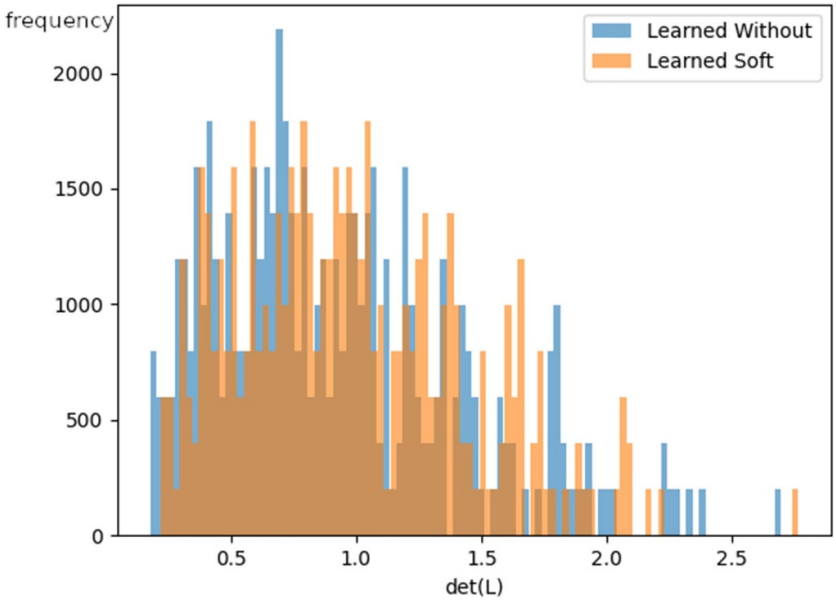


Figure 8. P2D: learned $\det(\mathbf{L})$.

sample trajectory, and figure 8 shows the distribution of learned $\det(\mathbf{L})$. The median determinant (after a normalization such that the determinant is equal to 1.0 in GT), was close to this value for both SJ and WJ, indicating a symplectic system.

3.5. Shivamoggi equations

Shivamoggi equations (32) represent a 4D Hamiltonian system that is not symplectic, and thus has a degenerate Poisson bivector. The equations were solved within the range of parameters $u \in [-0.5, 0.5]$, $x \in [-0.5, 0, 5]$, $y \in [-0.1, 0.1]$, $z \in [-0.5, 0.5]$. It was necessary to constraint the range of $\mathbf{r} = (u, x, y, z)$ because for instance when $u = 0$, the solutions explode [41]. Figure 9 shows the u-component over a sample trajectory. The initial conditions were sampled uniformly (500 points) as well as the GT (also 500 points), and each trajectory consisted of 150 steps.

We used 80 epochs in the learning, and results of the WJ and SJ methods are in figures 9 and 10. Figure 10 shows the distribution of $\log_{10}(\det(\mathbf{L}))$, indicating that the system is indeed degenerate. In comparison with determinants of $\mathbf{L}$ learned in the symplectic case of a 2D particle (P2D), see table 1, the learned determinants are quite low in the Shivamoggi case (after the same normalization as in the P2D case). Therefore, DPNNs are able to distinguish between symplectic and non-symplectic Hamiltonian systems.

3.6. Particle in 3D (P3D)

Motion of a P3D was simulated and learned with the same parameters as the P2D (section 3.4). Figure 11 shows momentum during a sample trajectory of a P3D space taken from the GT

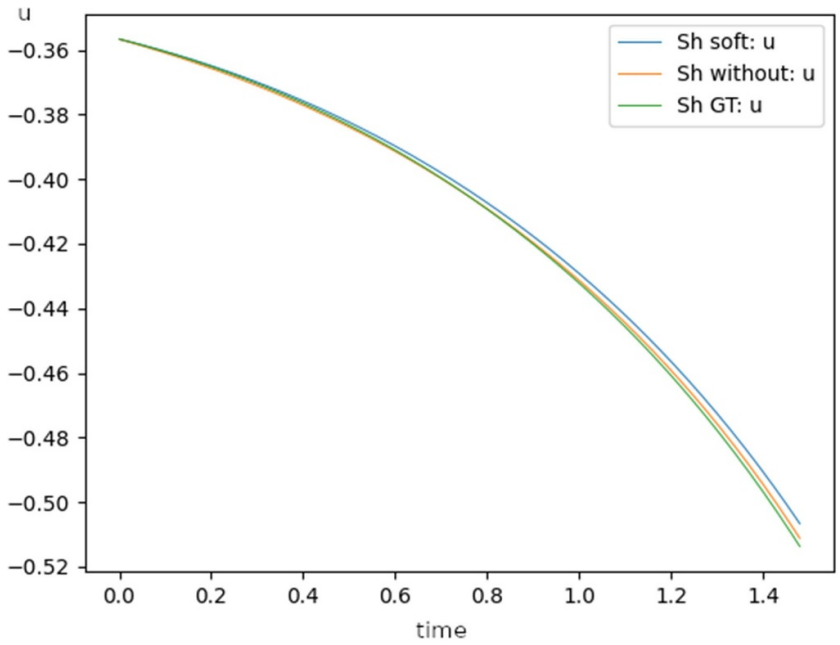


Figure 9. Shivamoggi: a sample trajectory, component u.

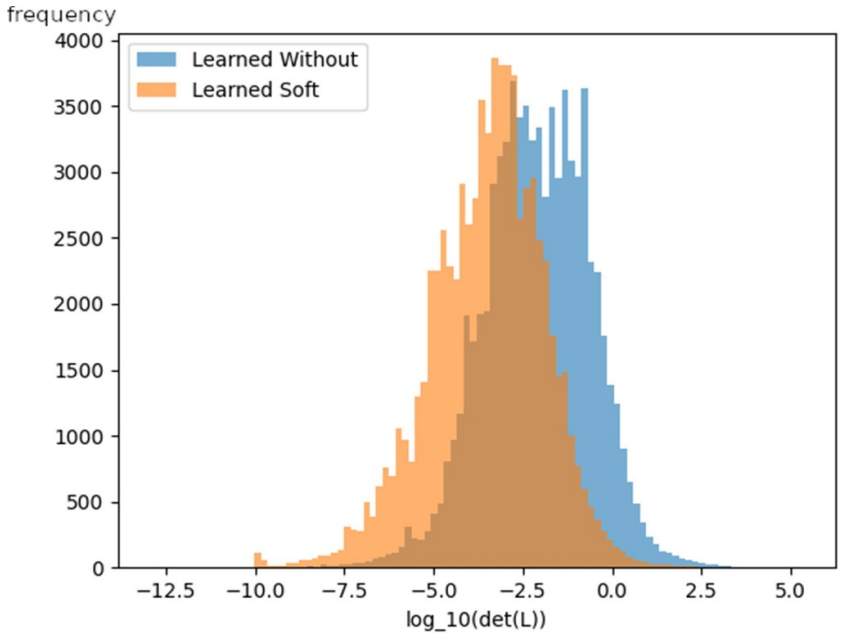


Figure 10. Shivamoggi: learned $\log_{10}(\det(\mathbf{L}))$.

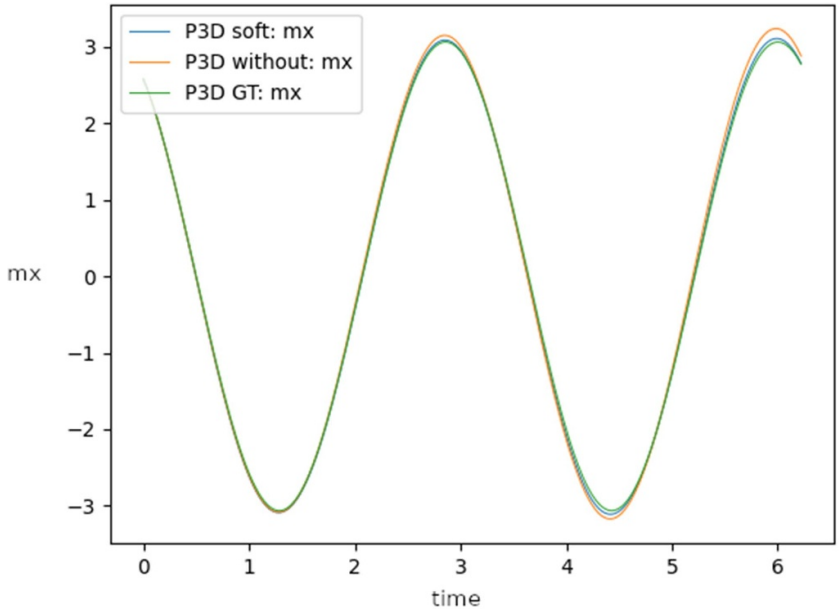


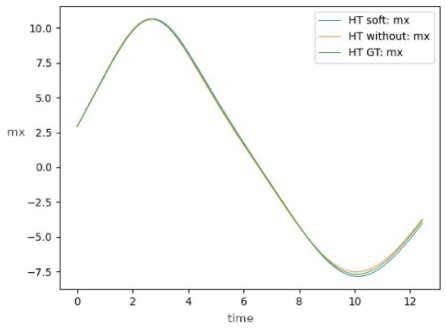
Figure 11. P3D: comparison of momentum $\mathbf{M}(t)$ on an exact trajectory (GT) and trajectories obtained by integrating the learned models (without Jacobi and with soft Jacobi) in the case of a 3D harmonic oscillator.

set, as well as trajectories with the same initial conditions obtained by DPNNs (WJ and SJ flavors). Training (and validation) dataset and GT dataset consisted of 200 and 400 trajectories, respectively. Table 1 contains numerical values of the learning errors. Both WJ and SJ learn the Poisson bivector as well as the trajectories (and thus also the energy) well. The median determinant is close to unity, which indicates a symplectic system.

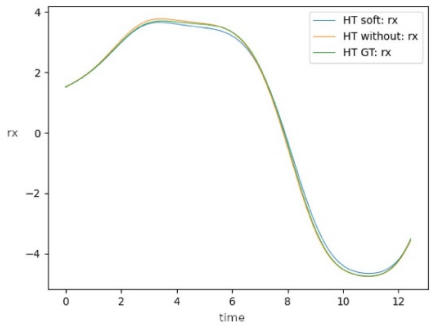
3.7. Heavy top (HT)

The HT dynamics is 6D, as well as the P3D. Parameters of the HT were the same as those of RB (section 3.3), except for taking 300 sampling points for the training and validation data, both c_{mov} and c_{Jac} were set to 100, number of epochs was 100, and constant Mgl was set to 0.981.

Figures 12(a) and (b) show a sample trajectory of a HT from the GT set and trajectories with the same initial conditions obtained by DPNNs. The training and validation were done in two sets of trajectories (with 300 and 400 trajectories, respectively). Numerical values of the learning errors can be found in table 1. For instance, the $\mathbf{L}$ matrix is close to being singular, indicating a non-symplectic system, but SJ learns slightly better than WJ. Similarly, as in the four-dimensional case, DPNNs distinguish between symplectic (P3D) and non-symplectic cases (HT).



(a) Angular momentum $\mathbf{M}(t)$ during a sample trajectory.



(b) Vector $\mathbf{r}(t)$ during the same trajectory.

Figure 12. Heavy top: comparison on an exact trajectory (GT) and trajectories obtained by integrating the learned models (without Jacobi and with soft Jacobi) in case of the heavy top.

4. Learning non-Hamiltonian systems

Let us now try to apply the WJ, SJ, and IJ methods, that are developed for learning purely Hamiltonian systems, to a non-Hamiltonian system, specifically a dissipative RB. A way to formulate the dissipative evolution of an RB is called the energetic Ehrenfest regularization [56], where the Hamiltonian evolution of an RB is supplemented with dissipative terms that keep the magnitude of angular momentum constant while dissipating the energy. The evolution equations are

$$\dot{\mathbf{M}} = \mathbf{M} \times E_{\mathbf{M}} - \frac{\tau}{2} \boldsymbol{\Xi} \cdot E_{\mathbf{M}} \tag{40}$$

where τ is a positive dissipation parameter and where $\boldsymbol{\Xi} = \mathbf{L}^T \cdot d^2E \cdot \mathbf{L}$ is a positive symmetric definite matrix (assuming that energy be positive definite) constructed from the Poisson bivector of the RB $L^{ij} = -\epsilon^{ijk}M_k$ and energy $E(\mathbf{M})$. These equations satisfy that $\dot{\mathbf{M}}^2 = 0$ while $\dot{E} \leqslant 0$, and their solutions converge to pure rotations around the principal axis of the RB (an axis with the highest moment of inertia), which is the physically relevant solution.

Results of learning trajectories generated by solving equation (40) are shown in figure 13. Parameters were the same as in section 3.3, except for taking the forward-Euler time-stepping, 60 epochs, 200 points for GT, and relaxation time $2 \cdot dt$ (dt being the simulation step, calculated as approximately 1% of the period). All the methods (WJ, SJ, and IJ) are capable to learn the trajectories to some extent, but WJ is the most successful, followed by SJ and IJ. As SJ, and especially IJ, use deeper properties of Hamiltonian systems (soft and exact validity of Jacobi identity), they are less robust in the case of non-Hamiltonian systems.

Figure 13 can be actually seen as an indication of non-Hamiltonianity of equation (40). Systems where IJ learns best, followed by SJ and WJ much more likely to be Hamiltonian, in contrast with non-Hamiltonian systems where WJ learns best, followed by SJ and IJ. In other words, DPNNs can distinguish between Hamiltonian and non-Hamiltonian systems by the order in which the flavors of DPNNs perform.

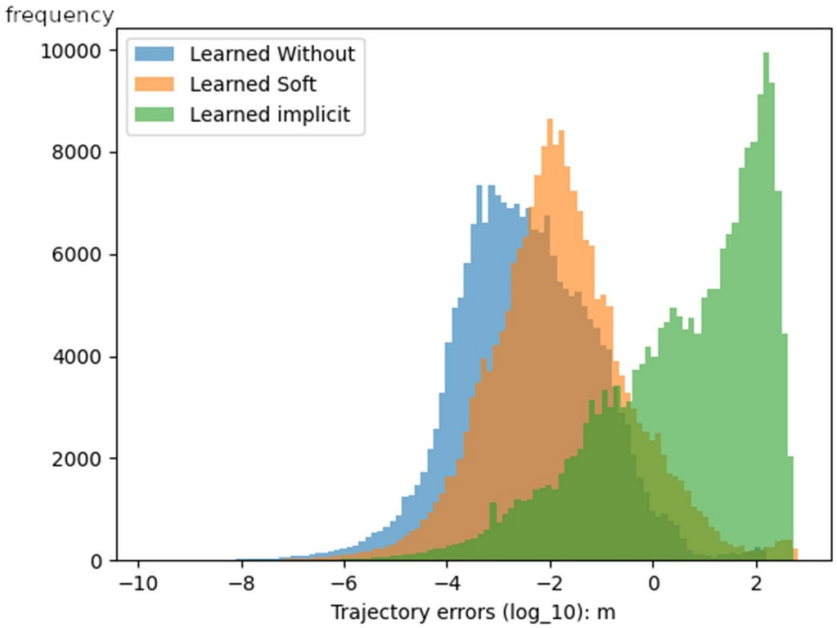


Figure 13. Distribution of errors in the angular momentum $\mathbf{M}$ when learning dissipative rigid-body dynamics (40) by methods assuming purely Hamiltonian systems (WJ, SJ, and IJ). WJ method is the most robust, capable to learn also the dissipative system relatively well (although worse than in the purely Hamiltonian case). The SJ method, which moreover softly imposes Jacobi identity, is less capable to learn the dissipative system. The IJ method, which has the best performance in purely Hamiltonian systems, see table 1, has the worst learning capability in the dissipative case.

5. Interpretation of results

The method and the insight gained contribute to the machine learning of equations research in the following ways.

1. Learning of Hamiltonian mechanics has been a popular topic in equation identification and learning. With, to our best knowledge, the only exception in [22, 24], the systems investigated were of symplectic nature. Improving understanding of non-symplectic systems is a pivotal contribution of this work since the development in this area may serve as a basis for more sophisticated learning procedures on such systems [57]. For example, [6] could now be used on complicated rotating systems with internal momenta, and symbolic expressions could be used to approximate scalar neural networks identified in our work.
2. We further demonstrate the usefulness of inductive biases in the learning of physics. Multiple works such as [58, 59] demonstrate the usefulness of inductive biases in the construction of Hamiltonian neural networks. We introduce new choices of such inductive biases and further solidify the belief that such modifications of the neural networks are crucial for the implementation of adequate models for physics-oriented tasks.

3. We propose a novel way how to distinguish Hamiltonian symplectic and non-symplectic systems, as well as dissipative and purely Hamiltonian evolutions. The approach of including dissipation in Hamiltonian neural networks is well known [60]. In principle, one would be able to learn this dissipation by the magnitude of the learned term and judge its contribution. However, our approach uses strictly Hamiltonian evolution and presents a way of extracting information about dissipativeness in a simple and direct way, just by comparing to what degree Jacobi identity is satisfied.

6. Conclusion

This paper proposes a machine learning method for learning Hamiltonian systems from data. DPNNs learn directly the Poisson bivector and Hamiltonian of the mechanical systems with no further assumptions about the structure of the systems. In particular, DPNN can distinguish between symplectic and non-symplectic systems by measuring the determinant of the learned Poisson bivector.

DPNNs come in three flavors: (i) without Jacobi identity (WJ), (ii) with softly imposed Jacobi identity (SJ), and with implicitly valid Jacobi identity (IJ). Although all the methods are capable to learn the dynamics, only SJ and IJ satisfy also the Jacobi identity. The typical behavior is that IJ learns Hamiltonian models most precisely, see table 1, followed by SJ and WJ.

When the three flavors of DPNNs are applied to learn a non-Hamiltonian system, it is expected that the order of precision gets reversed, making WJ most precise, followed by SJ and IJ. This reversed order of precision signalizes that the system learned by DPNNs is non-Hamiltonian rather than Hamiltonian.

In future, we would like to extend DPNNs to systems with dissipation prescribed by gradient dynamics [57, 61].

Data availability statement

The data that support the findings of this study will be openly available following an embargo at the following URL/DOI: https://github.com/enaipi/direct-Poisson-neural-networks.

Acknowledgment

MŠ and MP were supported by Czech Science Foundation, project 23-05736S. MP is a member of the Nečas center for Mathematical Modelling.

Reproducibility

Entire codebase necessary to reproduce this work is available publicly on Github https://github.com/enaipi/direct-Poisson-neural-networks under an open source MIT license.

ORCID iDs

Michal Pavelka https://orcid.org/0000-0003-0605-6737
Oğul Esen https://orcid.org/0000-0002-6766-0287
Miroslav Grmela https://orcid.org/0000-0001-7777-490X

References

[1] Einstein A 1936 Lens-like action of a star by the deviation of light in the gravitational field *Science* **84** 506–7
[2] Dermott S F, Murray C D, James J F, Gold T and Donnison J R 1984 Distribution and evolution of asteroid rotation rates [and discussion] *Phil. Trans. R. Soc.* A **313** 157–64
[3] DiPietro D M and Zhu B 2022 Symplectically integrated symbolic regression of Hamiltonian dynamical systems (arXiv:2209.01521)
[4] Karniadakis G E, Kevrekidis I G, Lu L, Perdikaris P, Wang S and Yang L 2021 Physics-informed machine learning *Nat. Rev. Phys.* **3** 422–40
[5] Raissi M, Perdikaris P and Karniadakis G E 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707
[6] Cranmer M D, Sanchez-Gonzalez A, Battaglia P W, Xu R, Cranmer K, Spergel D N and Ho S 2020 Discovering symbolic models from deep learning with inductive biases *CoRR* (arXiv:2006.11287)
[7] Lie S 1893 *Vorlesungen über Continuierliche Gruppen (in German)* (B. G. Teubner)
[8] Poincaré H 1901 Sur une forme nouvelle des équations de la méchanique *C.R. Acad. Sci.* **132** 369–71
[9] Abraham R and Marsden J E 1978 *Foundations of Mechanics* (Benjamin/Cummings Publishing Co., Inc.)
[10] Arnol'd V I 1989 *Mathematical Methods of Classical Mechanics* (*Graduate Texts in Mathematics* vol 60) 2nd edn (Springer) (translated from the Russian by K Vogtmann and A Weinstein)
[11] Libermann P and Marle C 1987 *Symplectic Geometry and Analytical Mechanics* (*Mathematics and its Applications* vol 35) (D. Reidel Publishing Co.) (translated from the French by Bertram Eugene Schwarzbach)
[12] Marsden J E and Ratiu T S 1999 *Introduction to Mechanics and Symmetry* (*Texts in Applied Mathematics* vol 17) 2nd edn (Springer)
[13] González D, Chinesta F and Cueto E 2019 Thermodynamically consistent data-driven computational mechanics *Contin. Mech. Thermodyn.* **31** 239–53
[14] Moya B, Gonzalez D, Alfaro I, Chinesta F and Cueto E 2019 Learning slosh dynamics by means of data *Comput. Mech.* **64** 511–23
[15] Šípka M and Pavelka M 2021 Learning the GENERIC evolution (arXiv:2109.12659)
[16] Zhong Y D, Dey B and Chakraborty A 2019 Symplectic ODE-Net: learning Hamiltonian dynamics with control (arXiv:1909.12077)
[17] Dierkes E and Flaßkamp K 2021 Learning Hamiltonian systems considering system symmetries in neural networks *IFAC-PapersOnLine* **54** 210–6
[18] Greydanus S, Dzamba M and Yosinski J 2019 Hamiltonian neural networks *Advances in Neural Information Processing Systems* vol 32, ed H Wallach, H Larochelle, A Beygelzimer, F D Alché-Buc, E Fox and R Garnett (Curran Associates, Inc.)
[19] Cranmer M, Greydanus S, Hoyer S, Battaglia P, Spergel D and Ho S 2020 Lagrangian neural networks (arXiv:2003.04630)
[20] Vaisman I 1994 *Lectures on the Geometry of Poisson Manifolds* (*Progress in Mathematics* vol 118) (Birkhäuser Verlag)
[21] Weinstein A 1998 Poisson geometry *Differ. Geom. Appl.* **9** 213–38
[22] Finzi M, Alexander Wang K and Wilson A G 2020 Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints *Advances in Neural Information Processing Systems* vol 33, ed H Larochelle, M Ranzato, R Hadsell, M F Balcan and H Lin (Curran Associates, Inc.) pp 13880–9
[23] Marsden J E and Ratiu T S 2013 *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems* (*Texts in Applied Mathematics*) (Springer)
[24] Jin P, Zhang Z, Kevrekidis I G and Karniadakis G E 2022 Learning Poisson systems and trajectories of autonomous systems via Poisson neural networks *IEEE Trans. Neural Netw. Learn. Syst.* **34** 8271–83
[25] Fecko M 2006 *Differential Geometry and Lie Groups for Physicists* (Cambridge University Press)
[26] Weinstein A 1983 The local structure of Poisson manifolds *J. Differ. Geom.* **18** 523–57
[27] Esen O, Ghose Choudhury A G and Guha P 2016 Bi-Hamiltonian structures of 3D chaotic dynamical systems *Int. J. Bifurcation Chaos Appl. Sci. Eng.* **26** 1650215

[28] Esen O, Guha P and Gümral H 2022 3*D*-flows generated by the curl of a vector potential and Maurer–Cartan equations *Turk. J. Math.* **46** 3234–44
[29] Gümral H and Nutku Y 1993 Poisson structure of dynamical systems with three degrees of freedom *J. Math. Phys.* **34** 5691–723
[30] Gümral H 2010 Existence of Hamiltonian structure in 3D *Adv. Dyn. Syst. Appl.* **5** 159–71
[31] Hernández-Bermejo B 2001 New solutions of the Jacobi equations for three-dimensional Poisson structures *J. Math. Phys.* **42** 4984–96
[32] Hernández-Bermejo B 2001 One solution of the 3D Jacobi identities allows determining an infinity of them *Phys. Lett.* A **287** 371–8
[33] Hernández-Bermejo B 2007 New solution family of the Jacobi equations; characterization, invariants and global Darboux analysis *J. Math. Phys.* **48** 022903
[34] Esen O and Guha P 2020 On the quest for generalized Hamiltonian descriptions of 3D-flows generated by the curl of a vector potential *Int. J. Geom. Methods Mod. Phys.* **17** 2050042
[35] Gümral H 1992 Bi-Hamiltonian structure of *N*-component Kodama equations *J. Phys. A: Math. Gen.* **25** 5141–9
[36] Gümral H and Nutku Y 1994 Bi-Hamiltonian structures of d-Boussinesq and Benney–Lax equations *J. Phys. A: Math. Gen.* **27** 193–200
[37] Arnold V I 1966 Sur la géometrie différentielle des groupes de lie de dimension infini et ses applications dans l'hydrodynamique des fluides parfaits *Ann. Inst. Fourier* **16** 319–61
[38] Landau L D and Lifshitz E M 1976 *Mechanics* (Butterworth-Heinemann)
[39] Esen O, Choudhury A G, Guha P and Gümral H 2016 Superintegrable cases of four-dimensional dynamical systems *Regul. Chaotic Dyn.* **21** 175–88
[40] Gonera C and Nutku Y 2001 Super-integrable Calogero-type systems admit maximal number of Poisson structures *Phys. Lett.* A **285** 301–6
[41] Guha P and Choudhury A G 2014 First integrals and Hamiltonian structure for a system of ordinary differential equations occurring in magnetohydrodynamics *AIP Conf. Proc.* **1582** 116–23
[42] Shivamoggi B K 1999 Current-sheet formation near a hyperbolic magnetic neutral line in the presence of a plasma flow with a uniform shear-strain rate: an exact solution *Phys. Lett.* A **258** 131–4
[43] Holm D D, Marsden J E and Ratiu T S 1998 The Euler–Poincaré equations and semidirect products with applications to continuum theories *Adv. Math.* **137** 1–81
[44] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Netw.* **2** 359–66
[45] Montúfar G, Pascanu R, Cho K and Bengio Y 2014 On the number of linear regions of deep neural networks *Proc. 27th Int. Conf. on Neural Information Processing Systems (NIPS '14)* vol 2 (MIT Press) pp 2924–32
[46] Telgarsky M 2017 Neural networks and rational functions *Proc. 34th Int. Conf. on Machine Learning* (*Proc. Machine Learning Research* vol 70) ed D Precup and Y W Teh (PMLR) pp 3387–93
[47] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436–44
[48] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (MIT Press)
[49] Dugas C, Bengio Y, Bélisle F, Nadeau C and Garcia R 2000 Incorporating second-order functional knowledge for better option pricing *Proc. 13th Int. Conf. on Neural Information Processing Systems (NIPS '00)* (MIT Press) pp 451–7
[50] Glorot X, Bordes A and Bengio Y 2011 Deep sparse rectifier neural networks *Proc. 14th Int. Conf. on Artificial Intelligence and Statistics (Fort Lauderdale, FL, USA, 11–13 April 2011)* (*Proc. Machine Learning Research* vol 15) ed G Gordon, D Dunson and M Dudík (PMLR) pp 315–23
[51] Rumelhart D E, Hinton G E and Williams R J 1986 Learning representations by back-propagating errors *Nature* **323** 533–6
[52] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[53] González D, Chinesta F and Cueto E 2019 Learning corrections for hyperelastic models from data *Front. Mater.* **6** 14
[54] Paszke A *et al* 2019 PyTorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* vol 32, ed H Wallach, H Larochelle, A Beygelzimer, F D Alché-Buc, E Fox and R Garnett (Curran Associates, Inc.) pp 8024–35
[55] Šípka M and Pavelka M 2023 Direct Poisson neural networks (GitHub Repository) (available at: https://github.com/enaipi/direct-Poisson-neural-networks)
[56] Pavelka M, Klika V and Grmela M 2019 Ehrenfest regularization of Hamiltonian systems *Physica* D **399** 193–210

[57] Cueto E and Chinesta F 2023 Thermodynamics of learning physical phenomena *Arch. Comput. Methods Eng.* **30** 4653–66
[58] Gruver N, Finzi M, Stanton S and Wilson A G 2022 Deconstructing the inductive biases of Hamiltonian neural networks (arXiv:2202.04836)
[59] Hernández Q, Badías A, Chinesta F and Cueto E 2023 Port-metriplectic neural networks: thermodynamics-informed machine learning of complex physical systems *Comput. Mech.* **72** 553–61
[60] Sosanya A and Greydanus S 2022 Dissipative Hamiltonian neural networks: learning dissipative and conservative dynamics separately *CoRR* (arXiv:2201.10085)
[61] Pavelka M, Klika V and Grmela M 2018 *Multiscale Thermo-Dynamics* (de Gruyter)