Journal of Computational and Applied Mathematics 417 (2023) 114608

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Learning Hamiltonians of constrained mechanical systems

Elena Celledoni, Andrea Leone, Davide Murari [*], Brynjulf Owren

Department of Mathematical Sciences, Norwegian University of Science and Technology, Norway

ARTICLE INFO

Article history:
Received 31 January 2022
Received in revised form 24 June 2022

Keywords:
Hamiltonian neural networks
Lie group integrators
Homogeneous manifolds
Hamiltonian systems
Constrained mechanical systems

ABSTRACT

Recently, there has been an increasing interest in modelling and computation of physical systems with neural networks. Hamiltonian systems are an elegant and compact formalism in classical mechanics, where the dynamics is fully determined by one scalar function, the Hamiltonian. The solution trajectories are often constrained to evolve on a submanifold of a linear vector space. In this work, we propose new approaches for the accurate approximation of the Hamiltonian function of constrained mechanical systems given sample data information of their solutions. We focus on the importance of the preservation of the constraints in the learning strategy by using both explicit Lie group integrators and other classical schemes.

1. Introduction

Neural networks have been proven to be effective in learning patterns from data in many different contexts. Recently there has been an increasing interest in applying neural networks to learn physical models from data, for example models of classical mechanics. For Hamiltonian systems, multiple approaches have been proposed to approximate the energy function, see, e.g., [1–5]. Building on these results, we propose an improved learning procedure. Our main contribution is an approach to learn the Hamiltonian for systems defined on the cotangent bundle $T^*\mathcal{Q}$ of some manifold $\mathcal{Q}$ embedded in a vector space. Under the assumption that $T^*\mathcal{Q}$ is homogeneous, we show how to do that while preserving the geometry during the learning phase. In this paper, by preservation of the geometry we mean the accurate conservation of the constraints rather than of other geometric features such as symplecticity, energy or other first integrals of the system.

As in [4], we express the dynamics of constrained systems by embedding the problem in a vector space of larger dimension, but in our approach we do not make use of Lagrange multipliers. With the aim of understanding the importance of the geometry in this approximation problem, we compare learning procedures based on numerical integrators that preserve the phase space of the system with others that do not. We restrict to homogeneous spaces where Lie group methods can preserve the geometry up to machine accuracy (see, e.g., [6]). For example, multi-body lumped mass systems fall naturally in this setting [7, Chapter 2]. This restriction still includes systems with the configuration manifold that is a Lie group, as in some problems of rigid body and rod dynamics, but we will not consider these applications here. The experiments show that there are specific problems where approximating the Hamiltonian using a Lie group method can be relevant. Surprisingly, in many other settings classical Runge–Kutta integrators produce comparable results.

The main focus of the present paper is to learn an approximation of a Hamiltonian system where the training data are given as a set of trajectory segments. To do so, one could learn the dynamics either by approximating the Hamiltonian vector field or the Hamiltonian function as done in our work. Another relevant difference in the learning framework

[*] Corresponding author.
E-mail address: davide.murari@ntnu.no (D. Murari).

https://doi.org/10.1016/j.cam.2022.114608

consists of considering in the training procedure either one time step of the flow map (see, e.g., [2]) or a sequence of successive time steps as proposed in [1]. In the latter work it is shown with experimental evidence that taking into account temporal dependencies improves performance. We follow the second strategy when dealing with unconstrained systems, whereas we test both of them with our approach to constrained systems.

In principle, the Hamiltonian can be any differentiable function. However, for mechanical systems, it is often made by the sum of (quadratic) kinetic energy and a potential energy, [8–10]. Following [3], we make the ansatz that the kinetic energy is characterized by a symmetric and positive definite matrix, and hence we aim to estimate it.

We conclude this Section with a more precise definition of the problem of interest. In the second Section, we introduce the Hamiltonian formalism for both unconstrained and constrained systems. In the third Section, we focus on unconstrained systems, presenting the general learning procedure that will be extended to constrained systems in the fourth Section. We also discuss how additional known information about the dynamical system can be included in the network training procedure. The experimental results show that physics-based regularization could be helpful to improve the extrapolation capability of the network and its stability in the presence of noise. In the last Section, we formalize the problem of learning a constrained Hamiltonian mechanical system and discuss the importance of the geometry for this class of problems. Finally, we complete this Section with numerical experiments in the `PyTorch` framework, showing how the predicted Hamiltonian depends on some training parameters and on the presence of noise. The numerical implementations are available in the GitHub repository associated to the paper[1].

1.1. Description of the problem

Suppose to be given a set of N sampled trajectories coming from a Hamiltonian system defined on a submanifold $\mathcal{M} = T^*\mathcal{Q}$ of $\mathbb{R}^{2n}$, where $T^*\mathcal{Q}$ is the cotangent bundle of the configuration manifold $\mathcal{Q}$ (see [11][Chapter 11] for more details). Moreover, assume that each of these trajectories contains M equispaced (in time) points. In other words, suppose that

$$\{(x_i, \bar{y}_i^2, \ldots, \bar{y}_i^M)\}_{i=1,\ldots,N}, \ \bar{y}_i^j = \Phi_{X_H}^{(j-1)\Delta t}(x_i) \tag{1}$$

as a training set, where $\Phi_{X_H}^t$ is the time t-flow of the exact, unknown Hamiltonian system. In practice, we never have access to the exact trajectories but to either a noisy version of them or a numerical approximation.

The approach we use aims to approximate the vector field $X_H \in \mathfrak{X}(\mathcal{M})$ that governs the dynamics, where by $\mathfrak{X}(\mathcal{M})$ we denote the collection of all smooth vector fields on $\mathcal{M}$. However, we know that such a vector field is Hamiltonian, i.e. there exists a scalar function $H : \mathcal{M} \to \mathbb{R}$ which, together with the geometry given by $\mathcal{M}$, characterizes the dynamics completely. For this reason, we do not need to directly approximate X_H, but just H and then eventually recover X_H.

The problem under consideration can be described as an inverse problem, since we want to infer the function H from trajectory data of the corresponding dynamical system rather than from samples of the function H itself. This description of the problem motivates how we measure the accuracy of our approximation, denoted by a parametric model H_Θ. Indeed, the target is not to approximate the trajectories of the given Hamiltonian system with some neural network, but to approximate the Hamiltonian. Thus the quality of the approximation can be computed in at least two ways. First, one can compare some measured trajectories with those obtained from the approximation. More precisely, we randomly generate $\tilde{N}$ initial conditions $z_i \in \mathcal{M}$, their $\tilde{M}$ time updates, and compute

$$\mathcal{E}_1\left(\left\{u_i^j\right\}_{i=1,\ldots,\tilde{N}}^{j=1,\ldots,\tilde{M}}, \left\{v_i^j\right\}_{i=1,\ldots,\tilde{N}}^{j=1,\ldots,\tilde{M}}\right) = \frac{1}{\tilde{N}\tilde{M}}\sum_{j=1}^{\tilde{M}}\sum_{i=1}^{\tilde{N}}\left\|u_i^j - v_i^j\right\|^2, \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm of $\mathbb{R}^{2n}$, $u_i^1 = z_i$, $v_i^1 = z_i$, $u_i^{j+1} = \Psi_{X_H}^h(u_i^j)$ and $v_i^{j+1} = \Psi_{X_{H_\Theta}}^h(v_i^j)$ for a numerical integrator Ψ^h of choice. One can randomly generate these initial conditions for academic examples where the true Hamiltonian is actually known. In this case, $\tilde{N}$ and $\tilde{M}$ can be specified arbitrarily, usually with $\tilde{N}$ less the number of training trajectories N. On the other hand, in more realistic applications one has to work with the initial conditions for which the related trajectory segments are known. In this case $\tilde{N}$ and $\tilde{M}$ are constrained by the available data, in particular the number of total trajectories is split into N for training and $\tilde{N}$ for test. In our experiments, we adopted the `SciPy` implementation of the Dormand–Prince pair of order (5,4) with a strict tolerance. In fact, following the PyTorch implementation of the mean squared error, $\mathcal{E}_1$ is actually divided by $2n$. Alternatively, as introduced in [12], one can compare pointwise values of the approximated and the true Hamiltonian, when known. This gives

$$\mathcal{E}_2(H, H_\Theta) = \frac{1}{\tilde{N}}\sum_{i=1}^{\tilde{N}}\left|H(z_i) - H_\Theta(z_i) - \frac{1}{\tilde{N}}\sum_{l=1}^{\tilde{N}}(H(z_l) - H_\Theta(z_l))\right|, \tag{3}$$

where $\mathcal{E}_2$ handles the fact that Hamiltonians differing, on $\mathcal{M}$, by a constant generate the same vector field. Indeed, $\mathcal{E}_2(H, H + c) = 0$.

[1] https://github.com/davidemurari/learningConstrainedHamiltonians

2. Hamiltonian mechanical systems

In this work, we focus on Hamiltonian mechanical systems based on a configuration manifold $\mathcal{Q} \subseteq \mathbb{R}^n$. We now introduce some basic elements of the theory of unconstrained Hamiltonian dynamics on $\mathbb{R}^{2n}$, which corresponds to the case $\mathcal{Q} = \mathbb{R}^n$. Then we extend this formulation to constrained systems on $T^*\mathcal{Q} \subset \mathbb{R}^{2n}$.

The Hamiltonian formalism gives a particular class of conservative vector fields which, in contrast to the Lagrangian one, can always be expressed with a system of first-order ordinary differential equations. For the unconstrained case, the equations are of the form $\dot{x}(t) = \mathbb{J}\nabla H(x(t)) := X_H(x(t))$ where $x(t) = [q(t), p(t)] \in \mathbb{R}^{2n}$ comprises the configuration variables and their conjugate momenta. Here, $H : \mathbb{R}^{2n} \to \mathbb{R}$ is a smooth function called the Hamiltonian of the system, and $\mathbb{J} \in \mathbb{R}^{2n\times 2n}$ is the symplectic matrix.

In this work, we focus on Hamiltonian systems whose energy function is of the form

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

where $M(q)$ is the mass matrix of the system, possibly depending on the configuration $q \in \mathbb{R}^n$, and $V(q)$ is the potential energy of the system. This is not a too restrictive assumption since it still includes a quite broad family of systems. For unconstrained systems, we will further restrict to the case where M is a constant matrix and the Hamiltonian is separable. This assumption allows to implement symplectic numerical integration without needing implicit updates. On the other hand, in the constrained setting we aim at preserving the geometry of the numerical flow map rather than other properties such as symplecticity. As a consequence, we can work with variable mass matrices still using explicit numerical integrators as in the unconstrained case.

We now briefly formalize how to extend this formulation to Hamiltonian systems that are holonomically constrained on some configuration manifold $\mathcal{Q} = \{q \in \mathbb{R}^n : g(q) = 0\}$ embedded in $\mathbb{R}^n$ (for a more detailed derivation of this formalism we refer to [7, Chapter 8]). Many mechanical systems relevant for applications are characterized by the presence of some constraints that are coupled to the ODE defining the dynamics. One way to model this kind of problems is based on Lagrange multipliers, which lead to differential algebraic equations (DAEs). There has been some work in the direction of extending the Hamiltonian neural network's framework to constrained systems (see, e.g., [4] in which this strategy of introducing Lagrange multipliers is applied).

In this manuscript, we want to present an alternative approach based on the assumption that the constrained manifold $\mathcal{Q}$ is embedded in some linear space $\mathbb{R}^n$. This is actually not a restriction, since Whitney's embedding theorem always guarantees the existence of such an ambient space (see, e.g., [11, Chapter 6]). More explicitly, because of this embedding property, constrained multi-body systems can be modelled by means of some projection operator and the vector field is written in such a way that it directly respects the constraints, without the addition of algebraic equations.

Furthermore, we assume that the components $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, are functionally independent on the zero level set, so that the Hamiltonian is defined on the $(2n - 2\,m)$ dimensional cotangent bundle $\mathcal{M} = T^*\mathcal{Q}$. Working with elements of the tangent space at q, $T_q\mathcal{Q}$, as vectors in $\mathbb{R}^n$, we introduce a linear operator that defines the orthogonal projection of an arbitrary vector $v \in \mathbb{R}^n$ onto $T_q\mathcal{Q}$, i.e.

$$\forall q \in \mathcal{Q}, \text{ we set } P(q) : \mathbb{R}^n \to T_q\mathcal{Q}, \quad v \mapsto P(q)v.$$

$P(q)^T$ can be seen as a map sending vectors of $\mathbb{R}^n$ into covectors in $T_q^*\mathcal{Q}$. If $g(q)$ is differentiable, assuming $G(q)$ is the Jacobian matrix of $g(q)$, we have $T_q\mathcal{Q} = \operatorname{Ker} G(q)$, and so $P(q) = I_n - G(q)\left(G(q)^T G(q)\right)^{-1} G(q)^T$, where $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix. This projection map allows us to define Hamilton's equations as follows

$$\begin{cases} \dot{q} = P(q)\partial_p H(q, p) \\ \dot{p} = -P(q)^T \partial_q H(q, p) + W(q, p)\partial_p H(q, p), \end{cases} \tag{4}$$

where

$$W(q, p) = P(q)^T \Lambda(q, p)^T P(q) + \Lambda(q, p)P(q) - P(q)^T \Lambda(q, p)^T,$$
$$\text{with} \quad \Lambda(q, p) = \frac{\partial P(q)^T p}{\partial q}.$$

It is important to remark that since $T^*\mathcal{Q} \subset \mathbb{R}^{2n}$, we can work with the coordinates of the ambient space in the subsequent development. We notice that when $\mathcal{Q} = \mathbb{R}^n$, we can set $P(q) = I$ and recover the unconstrained formulation. These equations of motion can be derived by the standard Hamilton's variational principle on the phase space or by the Legendre transform applied to the Euler–Lagrange equations. However, due to the geometry of the system, the variations need to be constrained to the right spaces and this is done with the projection map $P(q)$. We will focus on the case $Q = S^2 \times \cdots \times S^2 = (S^2)^k$ in Section 4.2, where the mass matrix $M(q)$ and Eq. (4) takes a structured form, with S^2 the unit sphere in $\mathbb{R}^3$.

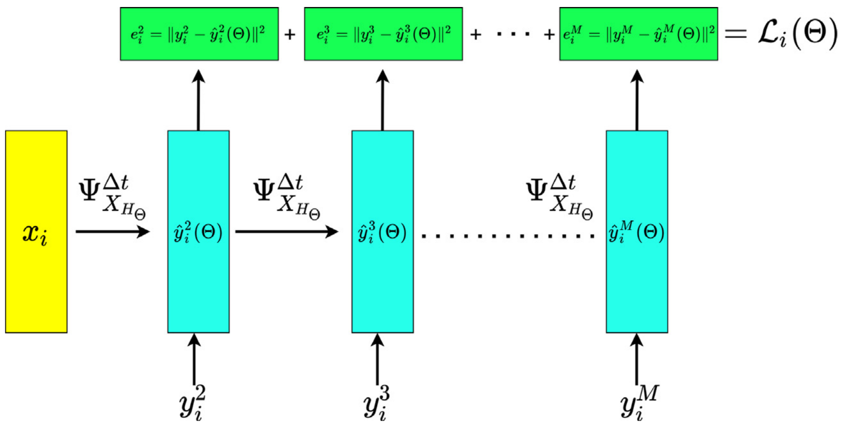

$e_i^2 = \|y_i^2 - \hat{y}_i^2(\Theta)\|^2$
$e_i^3 = \|y_i^3 - \hat{y}_i^3(\Theta)\|^2$
$e_i^M = \|y_i^M - \hat{y}_i^M(\Theta)\|^2$
$= \mathcal{L}_i(\Theta)$
x_i
$\Psi^{\Delta t}_{X_{H_\Theta}}$
$\hat{y}_i^2(\Theta)$
$\hat{y}_i^3(\Theta)$
$\hat{y}_i^M(\Theta)$
y_i^2
y_i^3
y_i^M

Fig. 1. Forward pass of an input training trajectory $(x_i, y_i^2, \dots, y_i^M)$. The picture highlights the resemblance to an unrolled version of a Recurrent Neural Network. The network outputs $(\hat{y}_i^2, \dots, \hat{y}_i^M)$.

3. Learning unconstrained systems

As in [1], we base the training on a recurrent approach, that is graphically described in Fig. 1.

As mentioned in Section 1.1, we work with numerically generated training trajectories that we denote by

$$\{(x_i, y_i^2, \dots, y_i^M)\}_{i=1,\dots,N}.$$

We limit the treatment of noisy training data to Section 3.2. To obtain an approximation of the Hamiltonian H, we define a parametric model H_Θ and look for a Θ so that the trajectories generated by H_Θ resemble the given ones. H_Θ in principle can be any parametric function depending on the parameters Θ. In our approach, Θ will collect a factor of the mass matrix and the weights of a neural network, as specified in Eq. (7). We use some numerical one-step method $\Psi^{\Delta t}_{X_{H_\Theta}}$ to generate the trajectories

$$\hat{y}_i^j(\Theta) := \Psi^{\Delta t}_{X_{H_\Theta}}(\hat{y}_i^{j-1}(\Theta)), \quad \hat{y}_i^1(\Theta) := x_i, \quad j = 2, \dots, M, \; i = 1, \dots, N. \tag{5}$$

For unconstrained problems we use symplectic numerical integrators, since they can take an explicit form and their adoption in the training procedure allows to have a target modified Hamiltonian to approximate (see, e.g., [13]). We then optimize a loss function measuring the distance between the given trajectories y_i^j and the generated ones $\hat{y}_i^j$, defined as

$$\mathcal{L}(\Theta) := \frac{1}{2n}\frac{1}{NM}\sum_{i=1}^{N}\mathcal{L}_i(\Theta) = \frac{1}{2n}\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\|\hat{y}_i^j(\Theta) - y_i^j\|^2, \tag{6}$$

where $\|\cdot\|$ is the Euclidean metric of $\mathbb{R}^{2n}$. This is implemented with the PyTorch `MSELoss` loss function. Such a training procedure resembles the one of Recurrent Neural Networks (RNNs), introduced in [14], as shown for the forward pass of a single training trajectory in Fig. 1. Indeed, the weight sharing principle of RNNs is reproduced by the time steps in the numerical integrator which are all based on the same approximation of the Hamiltonian, and hence on the same weights Θ. Finally, in Algorithm 1 we report one training epoch for a batch of data points.

3.1. Architecture of the network

In this work, the role of the neural network is to model the Hamiltonian, i.e. a scalar function defined on the phase space $\mathbb{R}^{2n}$. Thus, the starting and arrival spaces are fixed. For unconstrained systems we assume that

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

is separable. Here, the kinetic energy is a quadratic form defined by the symmetric positive definite matrix M^{-1}. It can hence be modelled through a learnable matrix A, $K(p) \approx K_A(p)$, by replacing M^{-1} or M with A^TA during the learning procedure. This modelling choice improves extrapolation properties since it allows to learn local (on a compact set) information that is valid on a larger domain, i.e. the mass matrix. In Section 4 we extend this reasoning to some configuration dependent mass matrices, where $M(q)$ is modelled through a constant symmetric and positive definite matrix. Recalling that A^TA can even be singular or close to singular, one can promote the positive definiteness of the modelled matrix adding a positive definite perturbation matrix to A^TA. Notice that, in principle, the imposition of

Algorithm 1 One epoch of the recurrent approximation of the Hamiltonian.

```
1: Choose a numerical integrator ($s$ stages)
2: $\hat{N} \leftarrow$ batch size,    Loss $\leftarrow 0$
3: for $i = 1, \dots, \hat{N}$ do
4:     $\hat{y}_i^1 \leftarrow x_i$
5:     for $j = 1, \dots, M$ do
6:         $\hat{y}_i^{j,[1]} \leftarrow \hat{y}_i^j$
7:         for $k = 1, \dots, s-1$ do
8:             Compute current value of Hamiltonian $H_\Theta(\hat{y}_i^{j,[k]})$
9:             Compute $\nabla H_\Theta(\hat{y}_i^{j,[k]})$                ▷ With automatic differentiation
10:            Compute stage $\hat{y}_i^{j,[k+1]}$
11:        end for
12:        Compute $\hat{y}_i^{j+1}$
13:        Increase Loss following equation (6)
14:    end for
15: end for
16: Optimize Loss
```

the positive (semi)definiteness of the matrix defining the kinetic energy is not necessary, but it allows to get more interpretable results. Indeed, it is known that the kinetic energy should define a metric on $\mathbb{R}^n$ and the assumption we are making guarantees such a property. For constrained systems we proceed in a similar way, as shown in Eq. (11). For the potential energy, a possible modelling strategy is to work with standard feedforward neural networks, and hence to define

$$V(q) \approx V_\theta(q) = f_{\theta_m} \circ \dots \circ f_{\theta_1}(q),$$

$$\theta_i = (W_i, b_i) \in \mathbb{R}^{n_i \times n_{i-1}} \times \mathbb{R}^{n_i},\ \ \theta := [\theta_1, \dots, \theta_m],$$

$$f_{\theta_i}(u) := \Sigma(W_i u + b_i),\ \ \mathbb{R}^n \ni z \mapsto \Sigma(z) = [\sigma(z_1), \dots, \sigma(z_n)] \in \mathbb{R}^n,$$

for example with $\sigma(x) = \tanh(x)$. In particular applications, where some additional information is known about the system, one can impose more structure on the architecture modelling $V(q)$. For example, in the case of odd potential or rotationally symmetric potential, one can define respectively an odd neural network V_θ or a rotationally equivariant one (see, e.g., [15]). Therefore, we have that

$$\Theta = [A, \theta], \quad H(q,p) \approx H_\Theta(q,p) = K_A(p) + V_\theta(q). \tag{7}$$

We remark that the Hamiltonian does not need to be approximated by a neural network, and hence in a compositional way. Many other parametrizations are possible. For example, starting from the sparse identification of dynamical systems approach presented in [16], in [17] it is proposed to parametrize the Hamiltonian with a dictionary of functions, for example polynomials and trigonometric functions. In our work, however, we opt for standard feedforward neural networks as the modelling assumption.

We now provide further details on the extrapolation capabilities of this network model. The learning procedure presented above is based on extracting temporal information coming from a set of trajectories belonging to a compact subset $\Omega \subset \mathbb{R}^{2n}$. In general, there is no reason why the Hamiltonian should be accurate outside of this set. To be more precise, denoting by $T > 0$ the largest time at which we know the trajectories, we have that

1. given enough samples in set Ω, distributed in order to capture the behaviour of the dynamical system, the prediction of the network is expected to be accurate in $\Omega_{[0,T]} := \{\Phi^t_{X_H}(x) : \ t \in [0,T],\ x \in \Omega\}$, i.e. for any $z_0 \in \Omega_{[0,T]}$ and any $\bar{t} > 0$ such that $\Phi^t(z_0) \in \Omega_{[0,T]}$ for all $t \in [0, \bar{t}]$,
2. outside $\Omega_{[0,T]}$ one cannot guarantee that the prediction will be accurate.

If we think of classical regression problems or even classification ones, it seems reasonable not to have information about the approximated quantity outside the sampled area. In those cases, with generalization we mean being sufficiently accurate close to the training points but still inside the sampled domain. However, here we know that the inferred function $H(q,p)$ has physical meaning and properties, so we might incorporate global known information about it to extend the applicability of the predictions.

This discussion supports the architectural choice for the kinetic energy suggested before (as in [3]). Indeed, supposing the Hamiltonian is separable, we know that the variable p appears in the energy function only via the quadratic form $\frac{1}{2}p^T M^{-1} p$. Thus, our modelling assumption allows us to approximate the mass matrix M just from a set of trajectories,

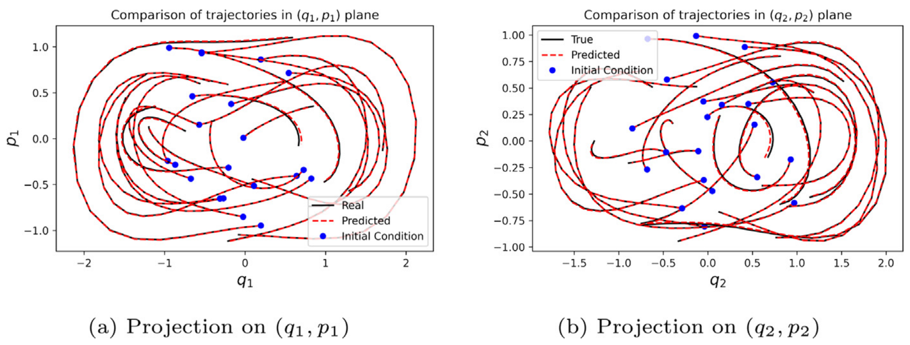


Fig. 2. Comparison of real and predicted test trajectories for the Hamiltonian (8). In this case, for the potential energy we used a feedforward network with 3 hidden layers having respectively 100, 50 and 50 neurons and tanh as activation function. The training integrator is Störmer–Verlet, with $M = 6$ and final time $T = 0.3$ and we use the Adam optimizer. The test trajectories, at $\tilde{M} = 20$ uniformly distributed points in the time interval [0, 1], are obtained with ODE(5,4). These trajectories correspond to $\tilde{N} = 100$ initial conditions on which the network has not been trained.

hence capturing the dependency of H on the variable p also outside $\Omega_{[0,T]}$. Other possible improvements can be obtained when some symmetry structure is known for the Hamiltonian. On a similar direction, in Section 3.2, we add some regularization based on other prior physical knowledge.

We present in Fig. 2 the comparison between ten learned trajectories and the corresponding exact ones of the Hamiltonian system $X_H \in \mathfrak{X}(\mathbb{R}^4)$ with Hamiltonian

$$H(q,p) = \frac{1}{2}\begin{bmatrix} p_1 & p_2 \end{bmatrix}^T \begin{bmatrix} 5 & -1 \\ -1 & 5 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \frac{q_1^4 + q_2^4}{4} + \frac{q_1^2 + q_2^2}{2}. \tag{8}$$

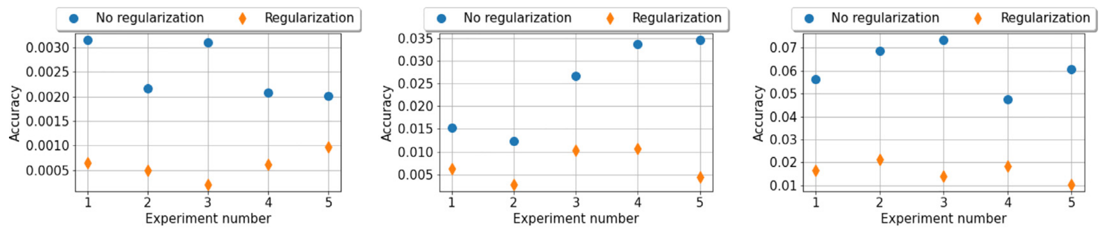
The training procedure of the network is based on 900 trajectories, sampled uniformly in 6 time instants, on the interval $0 \leq t \leq 0.3$. We remark that the training initial conditions are carefully chosen so that their associated trajectory segments well-capture the dynamics of interest. Fig. 2 collects test trajectories corresponding to the time interval [0, 1]. Since we are interested in approximating the Hamiltonian and not directly the trajectories, we are not constrained to evaluate the quality of the approximation with the same time integrator as the one used for training. In fact, these test trajectories have been generated with an embedded Runge–Kutta pair of order (5,4), with same relative and absolute accuracies for both the real and learned systems. Experimentally, it is clear that the qualitative behaviour of the Hamiltonian is well captured, as we can see from Fig. 2. To quantify the agreement of the prediction with the true Hamiltonian we report the $\mathcal{E}_1$ metric, as defined in (2), that is $6.59 \cdot 10^{-5}$. Furthermore, the training loss is $4.62 \cdot 10^{-7}$.

3.2. *Robustness to noise and regularization*

In real world applications, data is contaminated by noise which usually comes from the measurement process. Thus, we need to test the robustness of the learning framework to the presence of noise in the training trajectories. To do so, we synthetically generate the trajectories as before, and then add random normal noise to all the points except the initial condition (for an averaging strategy that allows to deal even with perturbed initial conditions, see, e.g., [1]). By construction, the network necessarily learns a Hamiltonian function, that is expected to generate trajectories close to the noisy ones. Since the training does not rely on clean trajectories, it is reasonable not to expect neither a loss value which is as small as in the absence of noise, nor a too accurate approximation of the Hamiltonian and the trajectories. Nevertheless, we aim for a learned Hamiltonian with level sets close to the exact ones, hence giving trajectories that resemble the true ones. One way of improving the quality of the neural networks proposed here, is to make use of a priori known physical properties of the dynamical system. We use an approach based on soft constraints which means that we take the known physical properties into account by adding a regularization term in the cost function. An example of such a property could be one or more known conserved quantities, so called first integrals. Hamiltonian systems always have at least one first integral, namely the Hamiltonian function itself, but there might be additional independent ones. Enforcing the first integrals to be preserved or nearly preserved seems to be a reasonable strategy for obtaining improved qualitative behaviour of the resulting approximation as shown in the following example.

Consider a Hamiltonian system with Hamiltonian function $H : \mathbb{R}^{2n} \to \mathbb{R}$, and a functionally independent first integral G, i.e. $\nabla H(x)$ and $\nabla G(x)$ are never parallel. Consider the numerical integration $\hat{y}_k^j$, $j = 1, \dots, M$ of the approximated Hamiltonian vector field X_{H_Θ}, starting at $\hat{y}_k^1 = x_k$. In the ideal case in which the learned Hamiltonian H_Θ coincides with H and the numerical flow is replaced with the exact one, both H and G should be conserved. For this reason, we suggest adding to the loss function in Eq. (6) the following "regularization" term:

$$\mu \sum_{j \in \mathcal{I}} \left(G(\hat{y}_k^j) - G(x_k) \right)^2$$



(a) Case $\varepsilon = 0.1$ (b) Case $\varepsilon = 0.3$ (c) Case $\varepsilon = 0.5$

Fig. 3. 5 repeated experiments for each perturbation regime. We plot on the y axis the average accuracy, in terms of the $\mathcal{E}_1$ measure, obtained with the trained network, when compared with the real (non-noisy) trajectories.

for all the training points x_k. Here $\mathcal{I}$ is a subset of indices contained in $\{1, \dots, M\}$, and μ is a regularization parameter that balances the importance of the preservation of the additional first integral against the perfect fitting of the training trajectories. We test this regularization procedure with the Hamiltonian system $X_H \in \mathfrak{X}(\mathbb{R}^4)$ defined by

$$H(q_1, q_2, p_1, p_2) = \frac{q_1^2 + p_1^2}{2} + \frac{p_2^2}{2} + \frac{1}{2}q_2^2 + \frac{1}{4}q_2^4 = h_1(q_1, p_1) + h_2(q_2, p_2).$$

This system has $G(q, p) := h_1(q_1, p_1)$ as an additional independent first integral other than H. We report in Fig. 3 some plots of the obtained $\mathcal{E}_1$ values as defined in (2). In these experiments we add some random noise of the form $\varepsilon\delta$ to the points y_i^j of the numerical trajectories, where $\delta \sim \mathcal{N}(0, 1)$ follows a standard normal distribution. The same experiment is run 5 times, and for each of these we plot the obtained $\mathcal{E}_1$ value. For each experiment we generate new training and test trajectories, and these are used for both the regularized training and the non regularized one. Furthermore, each experiment has a different random initialization of the weights, which is however shared between the regularized and non regularized networks. We notice that with regularization we can consistently get a better error in terms of the $\mathcal{E}_1$ measure. There is not a huge difference between the results, however. This suggests that when prior information is known, it might be important to experiment with this kind of regularizing terms. To conclude the Section, we highlight how remarkable it is that even without the regularization term, the trajectories are qualitatively well captured by the network and hence the test error is quite low. This is mostly due to the prior physical knowledge we impose on the learning procedure, i.e. that the vector field should be Hamiltonian. Indeed, since in the worst case the network approximates the wrong Hamiltonian, we always expect that it does not overfit the noisy trajectories, since they cannot be learned exactly. On the other hand, without the prior knowledge of the Hamiltonian nature of the system, all the overfitting problems of standard neural networks reoccur and the risk of being closer to an interpolant of the noisy trajectories is higher.

4. Learning constrained Hamiltonian systems

The approximation of the Hamiltonians of constrained mechanical systems with neural networks has already been studied in the literature. Two main approaches can be identified. One of them is based on local coordinates on the constrained manifold (see, e.g., [1,2]) and the other uses ambient space coordinates and Lagrange multipliers (see [4]). In principle, both the formulations apply to any constrained Hamiltonian system. However, as remarked in [4], the choice of a redundant system of coordinates usually gives a simpler expression for the Hamiltonian. This results in a more data efficient training procedure. In the second approach an embedded Runge–Kutta pair of order (5,4) is used to train the network. This choice inevitably leads to a drift from the constrained manifold during the training, even if it can be reduced by setting the tolerances of the integrator. However, in this way the cost of the integrator increases, hence this is not the most efficient way to preserve the constraints.

In this work, we use an alternative global formulation of the dynamics, as introduced in Section 2. In principle this formulation adapts to any constrained Hamiltonian system whose configuration manifold is a submanifold of $\mathbb{R}^n$. Coupling this description of the dynamics with the learning framework introduced in Section 3, their Hamiltonian functions can be approximated. To be more precise, one can use any numerical integrator to discretize the constrained trajectories and compare them with the training data. For example, Runge–Kutta 4 method can be used and this experimentally gives fast training procedures and accurate approximations of the Hamiltonian, as shown in the experiments of Section 4.3.

We remark that in general numerical integrators do not preserve the geometry of the system and there might be a drift from the constrained manifold (see, e.g., [9, Chapter 7]). Experimentally this does not seem to have a great impact on the quality of the predicted Hamiltonian in most of the cases. However, as we present in the numerical experiments with Lie group integrators, there might be situations in which one benefits from training the Hamiltonian with an integrator preserving the phase space. Notice that the Hamiltonian that defines the dynamics has non-unique extension outside the phase space $\mathcal{M} = T^*\mathcal{Q}$. This is due to the projection matrix $P(q) = I_n - G(q)\left(G(q)^T G(q)\right)^{-1} G(q)^T$ appearing Eq. (4), where $G(q)$ is the Jacobian matrix of the constraint function $g(q)$ defining $\mathcal{Q}$. This justifies investigating the importance of the preservation of the manifold T^*Q in the training procedure.

As introduced in Section 2, in this work we assume that the constrained configuration manifold $\mathcal{Q}$ is known. Referring to Eq. (4), we notice that once the geometry is known, it is enough to specify the Hamiltonian function $H : T^*\mathcal{Q} \subseteq \mathbb{R}^{2n} \to \mathbb{R}$ in order to characterize the dynamics of a system. We show a setting in which the geometry can be preserved by Lie group integrators (see, [6,18,19]) focusing on the case $T^*\mathcal{Q}$ is homogeneous.[2] We see this even as an opportunity to study the behaviour of this class of methods in an applied framework and combined with neural networks. This geometric setup applies, for example, when $\mathcal{Q}$ is a homogeneous manifold and the transitive action $\psi : G \times \mathcal{Q} \to \mathcal{Q}$ defines, for any $q \in \mathcal{Q}$, a submersion $\psi_q : G \to \mathcal{Q}$ at the identity element $e \in G$ (see, e.g., [20,21]). Cartesian products of homogeneous manifolds are homogeneous too. Usually, multibody systems have constrained configuration manifolds given by Cartesian products of S^2, $\mathbb{R}^k$, $SO(3)$ and $SE(3)$, which are respectively the special orthogonal and Euclidean groups. These are all homogeneous manifolds and so are their tangent and cotangent bundles.

4.1. Lie group methods and neural networks

Among the various classes of Lie group methods, we consider the Runge–Kutta–Munthe–Kaas (RKMK) methods and the commutator free ones (see, e.g., [22,23]). The underlying idea of RKMK methods, applied to $F \in \mathfrak{X}(\mathcal{M})$, with $\mathcal{M}$ an arbitrary homogeneous manifold, is to express F as $F|_m = \psi_*(f(m))|_m$. Here ψ_* is the infinitesimal generator of ψ, a transitive Lie group action of G on $\mathcal{M}$, and $f : \mathcal{M} \to \mathfrak{g}$ is a function that locally lifts the dynamics to the Lie algebra $\mathfrak{g}$ of G. On this linear space, we can perform a time step integration. We then map the result back to $\mathcal{M}$, and repeat this up to the final integration time. More explicitly, let Δt be the size of the uniform time step of the discretization, we then update $y_n \in \mathcal{M}$ to y_{n+1} by

$$\begin{cases} \gamma(0) = 0 \in \mathfrak{g}, \\ \dot{\gamma}(t) = \mathrm{dexp}^{-1}_{\gamma(t)} \circ f \circ \psi(\exp(\gamma(t)), y_n) \in T_{\gamma(t)}\mathfrak{g}, \\ y_{n+1} = \psi(\exp(\gamma_1), y_n) \in \mathcal{M}, \end{cases} \tag{9}$$

where $\gamma_1 \approx \gamma(\Delta t) \in \mathfrak{g}$ is computed with a Runge–Kutta method, and dexp^{-1} is the inverse of the differential of the exponential map $\exp : \mathfrak{g} \to G$ as defined, for example, in [18, Section 2.6]. We do not go into the details of commutator free methods, but the following development applies to them as well. In particular the function f still plays a fundamental role.

We now present a natural way to combine the learning framework typical of unconstrained systems with Lie group integrators. This is done introducing a Lie group method during the learning procedure. Indeed, since we want to apply a Lie group integrator to deal with nonlinear geometries, we set $\Psi^{\Delta t}$, defined in Eq. (5), to be the Δt update given by some RKMK method. In other words, using the notation of Eq. (9), we get $\Psi^{\Delta t}(z) = \psi(\exp(\gamma_1), z)$ with $\gamma_1 \in \mathfrak{g}$.

The setting presented above for generic vector fields on homogeneous manifolds simplifies considerably in the presence of Hamiltonian systems. Indeed, for this type of systems, what is needed to fully determine the dynamics is the geometry given by $\mathcal{M} = T^*\mathcal{Q}$ and the scalar Hamiltonian function $H : \mathcal{M} \to \mathbb{R}$. In other words, we can think of the function $f : \mathcal{M} \to \mathfrak{g}$, that allows to express the vector field in terms of the infinitesimal generator of the action, as the result of an operator $F : C^1(\mathcal{M}, \mathbb{R}) \to \{T^*\mathcal{M} \to \mathfrak{g}\}$ acting on a scalar function H. More explicitly, we can write $f = F[H]$ where F and H encode respectively the geometry and the dynamics of the system. This operator is not really necessary, but it clarifies considerably how the neural network comes into play in the learning framework. Indeed, because of this construction, we can write the numerical flow $\Psi^{\Delta t}$ as the map sending y_n into $y_{n+1} = \psi(\exp(\gamma_{\Delta t, y_n}), y_n)$ with $\gamma_{\Delta t, y_n}$ being an approximation of the solution $\gamma(\Delta t)$ of the following initial value problem

$$\begin{cases} \dot{\gamma}(t) = \mathrm{dexp}^{-1}_{\gamma(t)} \circ F[H_\Theta] \circ \psi(\exp(\gamma(t)), y_n) \in T_{\gamma(t)}\mathfrak{g}, \\ \gamma(0) = 0 \in \mathfrak{g}. \end{cases}$$

Here H_Θ is the approximation of the Hamiltonian given by the current weights Θ of the neural network. Thus, applying a particular family of geometric numerical integrators, we can directly study some constrained systems with the same ideas coming from learning unconstrained ones. Since following this procedure the geometry is preserved, one can consider replacing the Euclidean distance in the loss function defined in Eq. (6) with a Riemannian metric of the constrained manifold. This would bring to distances between points that correspond to the length of the minimal geodesic connecting them, which is in general different from the length of the segment in the ambient space having them as extrema. In the remaining part of the Section, we specialize this reasoning to mechanical systems defined on copies of T^*S^2. We focus on a chain of spherical pendula, but the geometric setting applies also to other systems (see, e.g., [7, Section 10.5]).

[2] A smooth manifold $\mathcal{M}$ is homogeneous if for any pair of points $m_1, m_2 \in \mathcal{M}$ there is $g \in G$ such that $\psi(g, m_1) = m_2$, where $\psi : G \times \mathcal{M} \to \mathcal{M}$ is a Lie group action. In other words, ψ is a transitive action.

4.2. Mechanical systems on $(T^*S^2)^k$

As anticipated in the introductory Section, in this geometric setting we are not involving symplectic integrators and we do not assume to have a separable Hamiltonian anymore. Thus, we now model a more general family of Hamiltonians as

$$H(q,p) = \frac{1}{2}p^TM^{-1}(q)p + V(q). \tag{10}$$

We model the potential energy as before, however we need an alternative strategy for the inverse of the mass matrix, which is no longer assumed to be constant. Based on the problem, one can choose various parametrizations of the mass matrix or its inverse. We decide to specialize the architecture based on the fact that the geometry of the system is known to be $\mathcal{M} = (T^*S^2)^k$, where $S^2 \subset \mathbb{R}^3$. We coordinatize $\mathcal{M}$ with $(q,p) = (q_1,\dots,q_k,p_1,\dots,p_k) \in \mathbb{R}^{6k}$. In this case, when $p \in \mathbb{R}^{3k}$ is intended as the vector of linear momenta, the matrix $M(q)$ in Eq. (10) is a block matrix, with

$$i,j = 1,\dots,k, \quad \mathbb{R}^{3\times 3} \ni M(q)_{ij} = \begin{cases} m_{ii}I_3, & i=j \\ m_{ij}(I_3 - q_iq_i^T), & \text{otherwise,} \end{cases}$$

see [7, Section 8.3.3] for further details. Here, the matrix having constant entries m_{ij} is symmetric and positive definite. For this reason, we leverage this form of the kinetic energy and learn a constant matrix $A \in \mathbb{R}^{k\times k}$ and a vector $b \in \mathbb{R}^k$ so that

$$\begin{bmatrix} m_{11} & \dots & m_{1k} \\ m_{21} & \dots & m_{2k} \\ \vdots & \vdots & \vdots \\ m_{k1} & \dots & m_{kk} \end{bmatrix} \approx A^TA + \begin{bmatrix} \tilde{b}_1 & 0 & \dots & 0 \\ 0 & \tilde{b}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \tilde{b}_k \end{bmatrix} \tag{11}$$

where $\tilde{b}_i := \max(0, b_i)$ are terms added to promote the positive definiteness of the right-hand side. We tested also elevating to the second power the b_i instead of taking the maximum with 0, but we got better results with the choice presented in Eq. (11). The matrix on the left-hand side of Eq. (11) is exactly the one appearing in Hamiltonian formulations with Cartesian coordinates, as the one used in [4].

For the spherical pendulum we have $k = 1$ and hence the Hamiltonian dynamics is defined on its cotangent bundle T^*S^2, which is a homogeneous manifold. This can be obtained thanks to the transitivity of the group action

$$\Psi : SE(3) \times T^*S^2 \to T^*S^2, \quad ((R,r),(q,p^T)) \mapsto (Rq, (Rp + r \times Rq)^T),$$

where the transpose comes from the usual interpretation of covectors as row vectors. As in [24, Chapter 6], we represent a generic element of the special Euclidean group $G = SE(3)$ as an ordered pair (R,r), where $R \in SO(3)$ is a rotation matrix and $r \in \mathbb{R}^3$ is a vector. With this specific choice of the geometry, the formulation presented in Eq. (4) simplifies considerably. Indeed $P(q) = I_3 - qq^T$ which implies $W(q,p) = pq^T - qp^T$. Replacing these expressions in (4) and using the triple product rule we end up with the following set of ODEs

$$\begin{cases} \dot{q} &= (I - qq^T)\partial_pH(q,p) \\ \dot{p} &= -(I - qq^T)\partial_qH(q,p) + \partial_pH(q,p) \times (p \times q). \end{cases} \tag{12}$$

This vector field $X(q,p)$ can be expressed as $\psi_*(F[H](q,p))(q,p)$ with

$$\psi_*((\xi,\eta))(q,p) = (\xi \times q, \xi \times p + \eta \times q), \quad (\xi,\eta) \in \mathfrak{g} = \mathfrak{se}(3)$$

and

$$F[H](q,p) = (\xi,\eta) = \left(q \times \frac{\partial H(q,p)}{\partial p}, \frac{\partial H(q,p)}{\partial q} \times q + \frac{\partial H(q,p)}{\partial p} \times p\right).$$

A similar reasoning can be extended to a chain of k connected pendula, and hence to a system on $(T^*S^2)^k$. The main idea is to replicate both the Eqs. (12) and the expression $F[H]$ for all the k copies of T^*S^2. A more detailed explanation can be found in [6].

We present in Fig. 4 the results obtained for the training of a double pendulum, i.e. $k = 2$. To train the network, we generate a set of $N = 500$ training trajectories with the embedded Runge–Kutta pair of order (5,4) of SciPy. The final integration time is $T = 0.1$ and $M = 5$. To model the potential energy, we use a feedforward network with 3 hidden layers of 100 neurons each. In the plots we show the configuration variables, $q_1,\ q_2 \in S^2$, obtained for 100 test trajectories in the time interval [0, 1], where the network H_Θ has been trained with a commutator free method of order 4.

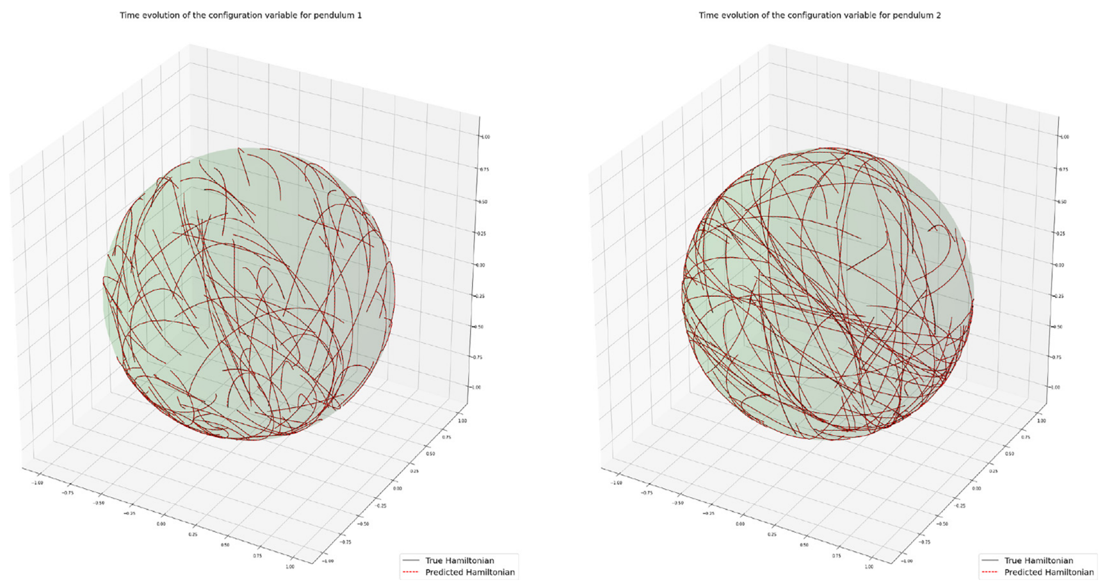


Fig. 4. Comparison between 100 test trajectories obtained with the true Hamiltonian H and the predicted one H_Θ. To train H_Θ, a Lie group method is used. This gives $\mathcal{E}_1 = 2.65 \cdot 10^{-6}$ and a final training loss of $1.6 \cdot 10^{-9}$.

Table 1
In this table we report the geometric means of the quantities $\mathcal{E}_1$, $\mathcal{E}_2$ and the training loss. Here we average over all the 240 experiments that have the same integrator. We denote the four integrators with EE (explicit Euler), LE (Lie Euler), RK4 (Runge–Kutta 4), and CF4 (commutator free 4).

Order	Integrator	$\mathcal{E}_1$	$\mathcal{E}_2$	Training loss
1	EE	5.7e−5	1.13e−2	2.12e−6
1	LE	4.9e−5	1.07e−2	1.17e−6
4	RK4	1.12e−5	3.83e−3	2.63e−7
4	CF4	1.12e−5	3.85e−3	2.64e−7

4.3. *Experimental study of the learning procedure*

We investigate the influence of the training setup on the error measures $\mathcal{E}_1$, $\mathcal{E}_2$, defined in (3), and on the training loss. More precisely, we test how the parameters M, N, the noise magnitude and the training integrator affect the performance of the network. We quantify the magnitude of noise in the training trajectories with a parameter $\varepsilon > 0$, as in Section 3.2. The integrators that we study are Lie Euler, explicit Euler (both of order 1), commutator free and Runge–Kutta (both of order 4). In particular, Lie Euler and commutator free methods preserve the phase space $\mathcal{M}$ up to machine accuracy. To get a sufficient sample of experiments, we repeat all the tests 5 times, and look at the medians and geometric means[3] of the obtained results. To be precise, we test $N \in \{50, 500, 1000, 1500\}$, $M \in \{2, 3, 5\}$, and $\varepsilon \in \{0, 0.001, 0.01, 0.1\}$. Therefore, we perform a total of 960 experiments, and also here the potential energy is modelled with a feedforward network of 3 hidden layers having 100 neurons each. Furthermore, for the four experiments performed varying just the integrator, and with the other parameters fixed, the network's weights are initialized to be the same, and also the training and test initial conditions are the same. For all these experiments, we focus on the single spherical pendulum, we keep the final training time to $T = 0.1$, and we do not use regularization terms. The training trajectories have been generated with the `SciPy` implementation of the Dormand–Prince pair of order (5,4) with strict tolerance.

As shown in Table 1, the order of the numerical integrator used to train the network plays an important role. Indeed, we get results that are similar for methods of the same order, but there is a noticeable decay in the errors and in the loss when we increase the order from one to four. As highlighted in [13], this effect can be explained with a standard argument of backward error analysis, see e.g. [9, Chapter 9]. From the results reported in Table 1 we see that the local error of the integrator is more important than the preservation of the geometry. Therefore, even if from a theoretical point of view it seems relevant to remain on the manifold during the training, in practice this does not seem to be very important in the particular experiment considered here. In Fig. 5, we plot the dependencies of $\mathcal{E}_1$, $\mathcal{E}_2$ and the training loss, on N, M, ε and the integrator. We notice that values of $\mathcal{E}_1$ below a threshold of 10^{-7} can be reached only

[3] The choice of geometric means is because of the exponential nature of the error measures and the training loss.

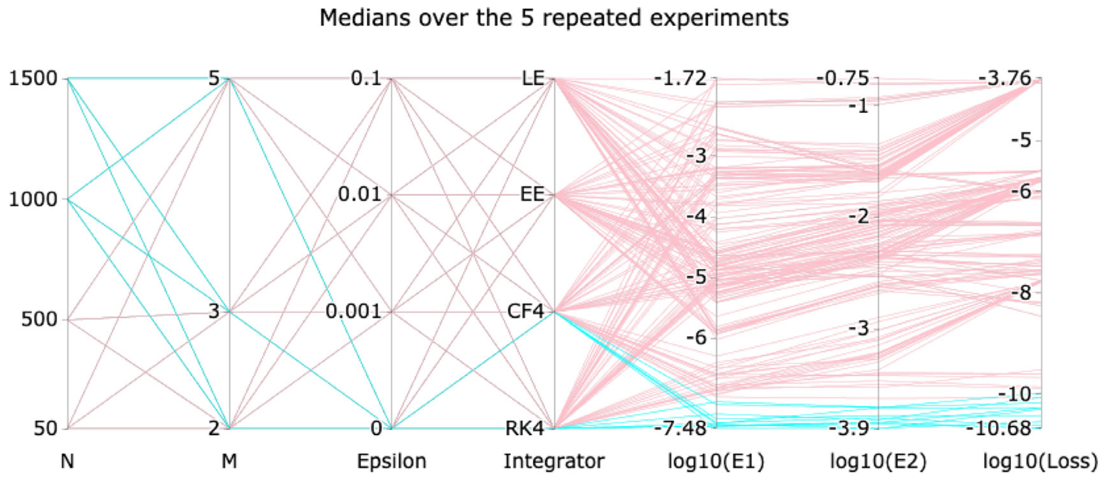


Fig. 5. This is a parallel coordinate plot reporting the dependencies of $\mathcal{E}_1$, $\mathcal{E}_2$ and the training loss on the parameters N, M, ε and on the integrator. Each coloured polyline corresponds to the median over 5 experiments, i.e. same N, M, ε and same integrator. The lines in cyan colour represent the combinations giving $\mathcal{E}_1 < 10^{-7}$.

Table 2
In this table we report the combinations that give the 5 best values of $\mathcal{E}_1$, together with the corresponding value $\mathcal{E}_2$. These are the geometric means among all the experiments. The two tables compare the performance on data with and without noise.

Without noise									
Integrator of order 1					Integrator of order 4				
N	M	Int.	$\mathcal{E}_1$	$\mathcal{E}_2$	N	M	Int.	$\mathcal{E}_1$	$\mathcal{E}_2$
1500	5	LE	1e-6	2.4e−3	1500	2	CF4	3.2e−8	1.3e−4
1000	5	LE	1e-6	2.3e−3	1500	5	RK4	3.3e−8	1.4e−4
1000	5	EE	1e-6	2.4e−3	1500	3	RK4	3.4e−8	1.4e−4
1500	5	EE	1e-6	2.5e−3	1500	2	RK4	3.6e−8	1.5e−4
500	5	LE	2e-6	2.6e−3	1500	3	CF4	3.7e−8	1.5e−4
With noise									
Integrator of order 1					Integrator of order 4				
N	M	Int.	$\mathcal{E}_1$	$\mathcal{E}_2$	N	M	Int.	$\mathcal{E}_1$	$\mathcal{E}_2$
1500	5	LE	1.2e−5	6.6e−3	1500	5	RK4	5e-6	3.4e−3
1500	5	EE	1.2e−5	6.3e−3	1500	5	CF4	6e-6	4.1e−3
1000	5	LE	1.5e−5	6.6e−3	1000	5	CF4	7e-6	3.8e−3
1000	5	EE	1.6e−5	6.8e−3	1000	5	RK4	8e-6	4.3e−3
500	5	LE	1.8e−5	6.7e−3	1000	3	RK4	8e-6	4.2e−3

with integrators of order four and with the smallest value of ε. The interplay of N, M and ε is further investigated in Table 2. An interactive version of Fig. 5, together with other parallel coordinate plots, can be found at the GitHub Page https://davidemurari.github.io/learningConstrainedHamiltonians/, while the dataset is available in the GitHub repository associated to the paper.

We conclude this parameter study considering separately the case with and without noise, $\varepsilon > 0$ and $\varepsilon = 0$ respectively. The results are reported in Table 2. In general the lowest values of $\mathcal{E}_1$ are obtained with high N. For the model under consideration, $N = 1000$ seems already high enough to achieve good results. Regarding M, Table 2 shows that to achieve lower values of $\mathcal{E}_1$ in the presence of noise, one needs to adopt a higher M. On the other hand, in the absence of noise it seems important to have a high M only for low order integrators. Finally, as may be expected, even if this Table does not distinguish among the different magnitudes of the noise, we see that with $\varepsilon = 0$ better results can be achieved.

We also point out that the experiments were performed for short integration times, where not only symplectic integrators can generate physically meaningful trajectories. It would be interesting to explore the performance of symplectic and constraint preserving integrators in this setting (see, e.g., [25]) and we defer this to further work.

Besides the theoretical aspect of the non-uniqueness of the extension of the dynamics outside of $\mathcal{M} \subset \mathbb{R}^{2n}$, we now report a numerical experiment where the preservation of the geometry during the training is beneficial. We consider again a simple spherical pendulum and we assume to know that the potential energy is linear. We hence impose this prior information on the architecture of the network. Due to the problem's simplicity, we aim to reach very low $\mathcal{E}_1$ and

Table 3
Comparison of the accuracy measures $\mathcal{E}_1$ and $\mathcal{E}_2$ obtained with the two integrators. These results are obtained imposing the linear structure of the potential energy on the network modelling the Hamiltonian of the spherical pendulum. The kinetic energy has been modelled as in previous experiments.

Numerical method in the training	$\mathcal{E}_1$	$\mathcal{E}_2$
Runge–Kutta of order 4	4.2e−12	1.5e−6
Commutator free of order 4	1.1e−14	2.5e−7

$\mathcal{E}_2$ values. Training the same architecture for 200 epochs, both with Runge–Kutta and commutator free methods of order 4, we get the results in Table 3. Indeed the geometric integrator outperforms the classical Runge–Kutta method in this experiment.

This experiment suggests that the choice of an integrator that does not fully exploit the available information, like the geometry, might limit the quality of the obtained approximations. For those cases in which one is interested in as accurate as possible predictions, this might be a relevant issue.

The experiments performed lead to the conclusion that modelling multi-body systems with neural networks can be a valuable approach. However, to better leverage the approximation capabilities of machine learning techniques (see, e.g., [26,27]) we believe that a deeper investigation and understanding of how they interface with physical models is necessary.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860124.

The authors are grateful to Ergys Çokaj for the valuable discussions in the early stage of this work. The authors would like to thank the Isaac Newton Institute for Mathematical Sciences, Cambridge, for support and hospitality during the programme "Mathematics of Deep Learning" in the second half of 2021.

References

[1] Z. Chen, J. Zhang, M. Arjovsky, L. Bottou, Symplectic recurrent neural networks, in: International Conference on Learning Representations, 2020.
[2] S. Greydanus, M. Dzamba, J. Yosinski, Hamiltonian Neural networks, Adv. Neural Inf. Process. Syst. 32 (2019) 15379–15389.
[3] Y.D. Zhong, B. Dey, A. Chakraborty, Symplectic ode-net: Learning hamiltonian dynamics with control, 2019, arXiv preprint arXiv:1909.12077.
[4] M. Finzi, A. Wang, A.G. Wilson, Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints, NeurIPS (2020).
[5] C. Offen, S. Ober-Blöbaum, Symplectic integration of learned Hamiltonian systems, 2021, arXiv preprint arXiv:2108.02492.
[6] E. Celledoni, E. Çokaj, A. Leone, D. Murari, B. Owren, Lie Group integrators for mechanical systems, Int. J. Comput. Math. (2021) 1–31.
[7] T. Lee, M. Leok, N.H. McClamroch, Global Formulations of Lagrangian and Hamiltonian Dynamics on Manifolds, in: Interaction of Mechanics and Mathematics, Springer, Cham, ISBN: 978-3-319-56951-2; 978-3-319-56953-6, 2018, p. xxvii+539, http://dx.doi.org/10.1007/978-3-319-56953-6, 3699476.
[8] E.T. Whittaker, A Treatise on the Analytical Dynamics of Particles and Rigid Bodies, Cambridge University Press, ISBN: 0521 35883 3, 1993, Fourth edition.
[9] E. Hairer, M. Hochbruck, A. Iserles, C. Lubich, Geometric numerical integration, Oberwolfach Rep. 3 (1) (2006) 805–882.
[10] J.E. Marsden, T.S. Ratiu, Introduction to mechanics and symmetry, Phys. Today 48 (12) (1995) 65.
[11] J.M. Lee, Introduction to Smooth Manifolds, in: Graduate Texts in Mathematics, Springer New York, NY, 2012.
[12] M. David, F. Méhats, Symplectic learning for Hamiltonian neural networks, 2021, arXiv preprint arXiv:2106.11753.
[13] A. Zhu, P. Jin, Y. Tang, Deep Hamiltonian networks based on symplectic integrators, 2020, arXiv preprint arXiv:2004.13830.
[14] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, Technical Report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
[15] E. Celledoni, M.J. Ehrhardt, C. Etmann, B. Owren, C.-B. Schonlieb, F. Sherry, Equivariant neural networks for inverse problems, Inverse Problems 37 (2021).
[16] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937.
[17] D.M. DiPietro, S. Xiong, B. Zhu, Sparse symplectically integrated neural networks, in: Advances in Neural Information Processing Systems, vol. 34, 2020.
[18] A. Iserles, H.Z. Munthe-Kaas, S.P. Nørsett, A. Zanna, Lie-Group methods, Acta Numer. 9 (2000) 215–365.
[19] E. Celledoni, H. Marthinsen, B. Owren, An introduction to Lie group integrators–basics, new developments and applications, J. Comput. Phys. 257 (2014) 1040–1061.
[20] R. Brockett, H. Sussmann, Tangent bundles of homogeneous spaces are homogeneous spaces, in: Proc. Amer. Math. Soc., 35, (2) 1972 pp. 550–551.
[21] E. Celledoni, E. Çokaj, A. Leone, D. Murari, B. Owren, Dynamics of the N-fold pendulum in the framework of Lie group integrators, 2021, arXiv preprint arXiv:2109.12325.
[22] H. Munthe-Kaas, High order Runge–Kutta methods on manifolds, Appl. Numer. Math. 29 (1999) 115–127.
[23] E. Celledoni, A. Marthinsen, B. Owren, Commutator-free Lie group methods, Future Gener. Comput. Syst. 19 (3) (2003) 341–352.
[24] D.D. Holm, Geometric Mechanics-Part II: Rotating, Translating and Rolling, World Scientific, 2011.
[25] H.C. Andersen, Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations, J. Comput. Phys. 52 (1) (1983) 24–34.
[26] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[27] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314.