



Hamiltonian neural networks for solving equations of motionMarios Mattheakis ^{1,*}, David Sondak,¹ Akshunna S. Dogra ^{1,2,3} and Pavlos Protopapas¹¹*John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts 02138, USA*²*Department of Mathematics, Imperial College London, London SW7 2AZ, United Kingdom*³*EPSRC CDT in Mathematics of Random Systems: Analysis, Modelling and Algorithms, London SW7 2AZ, United Kingdom*

(Received 2 February 2022; accepted 10 June 2022; published 30 June 2022)

There has been a wave of interest in applying machine learning to study dynamical systems. We present a Hamiltonian neural network that solves the differential equations that govern dynamical systems. This is an equation-driven machine learning method where the optimization process of the network depends solely on the predicted functions without using any ground truth data. The model learns solutions that satisfy, up to an arbitrarily small error, Hamilton's equations and, therefore, conserve the Hamiltonian invariants. The choice of an appropriate activation function drastically improves the predictability of the network. Moreover, an error analysis is derived and states that the numerical errors depend on the overall network performance. The Hamiltonian network is then employed to solve the equations for the nonlinear oscillator and the chaotic Hénon-Heiles dynamical system. In both systems, a symplectic Euler integrator requires two orders more evaluation points than the Hamiltonian network to achieve the same order of the numerical error in the predicted phase space trajectories.

DOI: [10.1103/PhysRevE.105.065305](https://doi.org/10.1103/PhysRevE.105.065305)**I. INTRODUCTION**

Studying the evolution of dynamical systems has become a significant trend in scientific research. The information age has generated an exponential increase in the amount of digital data being stored, and a nontrivial fraction of these data sets describe the evolution of dynamical systems. These include a wide range of systems, from large-scale astrophysics to nanoscale quantum physics. Recently, machine learning models, and particularly neural networks (NNs), have been used to explore such data sets and forecast the future behavior of complex dynamical systems [1–3], spatiotemporal chaotic behavior [4,5], classify time series [6,7], improve turbulence models [8–11], discover differential equations (DEs) [12–15], and find approximate solutions for those equations [16,17]. In addition to the data-driven studies, equation-driven and data-free unsupervised NNs have been used to solve ordinary and partial DEs relevant to a variety of physical systems [18–22]. Equation-driven networks construct analytical functions that satisfy a particular differential structure; subsequently, in the training process of such models, we do not need any ground truth data. Essentially, the loss function solely depends on the solutions obtained by the NN while the training process is fully data-free. This formulation results in an unsupervised learning method. We emphasize that the proposed method does not use any data generated by traditional numerical solvers. Furthermore, the universal approximation theorem of NNs [23] states that a NN can approximate any smooth function with arbitrary accuracy. This makes NNs a suitable approach to solving complicated problems governed by DEs.

Physics-inspired and physics-informed NNs have been widely used for solving DEs, providing some potential advantages over using traditional integrators [24]. The effectiveness of these machine-learning solvers have been demonstrated by tackling challenging problems, where traditional numerical methods become inefficient, like solving high-dimensional partial differential equations [19,20], systems with moving boundary [25], and inverse problems [17,26,27]. Solving DEs with NNs is a rapidly growing field and techniques are regularly proposed to advance and improve these machine-learning solvers including Monte Carlo sampling [22], Fourier neural operators [28], curriculum regularization and sequential learning [29]. This paper contributes to this effort by introducing a Hamiltonian structure in the NN framework that improves the solving capability of nonlinear Hamiltonian systems. The computations of a NN can be efficiently implemented on parallel architectures, leading to significant speedup [18]. Indeed, recent hardware innovations and, in particular, the wide adoption of and access to GPUs, can drastically accelerate the computation process with minimal parallelization effort. This is a great advantage over traditional integrators where time-parallel algorithms are challenging to develop and implement. An overview of advantages and challenges in parallel in time integration methods are summarized by Ref. [30], while Ref. [31] shows that modern methods have been invented to parallelize the time integration and can be used in deep networks for a layer-parallel training accelerating the network optimization.

Data-driven Hamiltonian NNs have been proposed to impose physically informed inductive biases in the learning process. These networks are trained faster and generalize better than regular fully connected NNs, while they learn and respect exact conservative quantities such as the energy

*mariosmat@seas.harvard.edu

[32–35]. More specifically, Greydanus *et al.* [32] introduced Hamiltonian networks with embedded an Hamiltonian formalism showing that a NN can be used to learn a Hamiltonian that describes some given temporal trajectories. The time derivatives and time dependence are eliminated by using Hamilton's equations and automatic differentiation resulting a time-invariant energy. Once the Hamiltonian has been learned, predicting the motion of different initial conditions within and outside the training regime is possible by numerically solving Hamilton's equations. Recently, this approach has been successfully applied to learn Hamiltonians, forecast chaotic behavior, and predict transition to chaos [34,36]. The framework of the Hamiltonian NNs is quite general and can be implemented in different machine-learning architectures like reservoir computing [37] and graph networks as well as numerical integrators, which can be embedded in the network architecture to provide further improvement in the long-term forecasting [35,38]. Moreover, generative Hamiltonian networks have been proposed to generate trajectories that respect certain underlying laws like energy and momentum conservation and, subsequently, the generated data respect fundamental physical principles [39]. Other extensions of standard Hamiltonian networks consider learning the dynamics of systems in the presence of external driven forces and dissipation. Adopting more general formulations like a port-Hamiltonian, NNs are capable of predicting trajectories for damped and driven time-varying dynamical systems as well as efficiently uncovering underlying physical quantities hidden in data, like a stationary Hamiltonian, dissipation parameters, and external time-dependent forces [40]. These recent studies evidence that the learning capability of NNs can be drastically improved by embedding a Hamiltonian formulation in the framework; nevertheless, the advantages of imposing Hamiltonian's equations in NNs to solve DEs have not been studied yet. In this paper, we introduce and investigate a Hamiltonian NN used to solve the equations of motion of nonlinear dynamical systems. This is an equation-driven approach instead of a data-driven model because the form of the Hamiltonian and the initial state of a system are assumed to be known, while ground truth trajectories (data) are not required in the training process. In other words, standard Hamiltonian networks learn the Hamiltonian function from given data, whereas, our proposed model discovers trajectories that approximately satisfy Hamilton's equations. Subsequently, the two approaches are conceptually different despite the fact that the Hamiltonian formulation is embedded in both networks.

The current paper presents a data-free Hamiltonian NN architecture used for solving DE systems. Despite the success of physics-inspired NNs in solving DEs, Hamiltonian NN solvers have not been explored yet. Subsequently, the proposed Hamiltonian NN is an evolution of previously used data-free NNs for approximating solutions to DEs that identically satisfy boundary and initial conditions. We improve upon other NN DE solvers by speeding up the convergence of the network to the solution while reaping the benefits of the underlying physical properties. We propose a NN architecture inspired by and geared toward Hamiltonian systems with time-independent Hamiltonians. Once optimized, the NN satisfies Hamilton's equations over the entire temporal domain, directly implying the conservation of every invari-

ant under the respective Hamiltonian flow. As discussed in Ref. [20], calculating second derivatives using automatic differentiation is much more expensive than the calculation of first derivatives. Here we avoid this bottleneck by solving systems of first order DEs, Hamilton's equations, instead of second-order equations. NN solvers are conceptually different than traditional numerical solvers. Symplectic integrators are designed to conserve the energy over long-time ranges. Being iterative solvers, these traditional methods accumulate errors in time and also require values of the calculations at previous time points to construct an approximate solution. Traditional integrators conserve a Hamiltonian (energy) that is slightly different than the true Hamiltonian. On the other hand, the suggested Hamiltonian NN evaluates each time point independently and simultaneously satisfies all the DEs of a system. As a result, the Hamiltonian network conserves the original Hamiltonian and leads to a significant reduction in any accumulated numerical errors. Another distinct machine-learning direction is the development of NN integrators [16,19]. These hybrid models combine traditional integrators with NNs improving the performance in solving DEs. Our NN solver does not belong to this class of machine learning methods since it does not require a structured mesh or embed any iteration algorithm. On the other, the proposed model suggests an alternative way to solve ordinary differential equations (ODEs) with NNs without embedding conventional integrators. The proposed Hamiltonian NNs consist of a more numerically precise and robust method to solve dynamical equations than standard semi-implicit schemes such as a symplectic Euler integrator. By sharing the network weights, choosing a trigonometric activation function, penalizing violations in energy conservation law, and using an efficient parametric form of solutions, we show a speedup in the convergence behavior during the optimizing process and, subsequently, an improvement in the predictability of the network. Also, we show that after training the proposed NN architecture can be considered a true and globally symplectic unit and thereby a time-invariant unit.

In the rest of this paper, we describe the Hamiltonian NN architecture that is used to approximate Hamiltonian trajectories. An error analysis is performed and shows that the accuracy of the predicted solutions can be predefined before optimizing the network. Then, the proposed symplectic NN is applied to solve the equations that describe the motion of a nonlinear oscillator and a two-dimensional chaotic system. We point out situations where the Hamiltonian NN solver outperforms the semi-implicit Euler numerical method, a first-order symplectic integrator. However, a comparison with higher order symplectic integrators is not presented in this paper. The network performance is demonstrated by exploring different architectures through different parametric solutions and activation functions. Accurate long-time solutions are obtained by using a regularization term to encourage the discovery of solutions that conserve the total energy. The experiments presented in this paper have been performed using PYTORCH [41] and the codes are published on GitHub [42]. We conclude this study with a summary of the key ideas introduced in this paper, the advantages of using a Hamiltonian NN to solving DEs, and with a discussion of future plans.

II. HAMILTONIAN NEURAL NETWORK

A. Network architecture

A cornerstone idea in classical mechanics is that a system's evolution can be investigated through the study of its underlying symmetries and constraints. By the 20th century, Lagrange, Hamilton, and others had shown that the dynamics of a system is tethered to simple scalar functions, the Lagrangian and Hamiltonian functions, with multiple conservation laws and their underlying symmetries prepackaged with these functions. These scalar functions are then used to derive the DEs that govern the motion of a system. In particular, starting from the Lagrangian (the difference between kinetic and potential energy), invoking Hamilton's principle (the motion follows trajectories that minimize the action integral), and employing techniques from the calculus of variations, the motion of a system is described by the Euler-Lagrange (E-L) equations. In the Hamiltonian formulation, on the other hand, we start from the Hamiltonian which is a transformation of the Lagrangian and is a conservative quantity, namely, it does not change in time. This formulation results in Hamilton's equations, which are equivalent to the E-L equation and therefore minimize the same action. Hamilton's equations are a coupled set of first-order DEs, whereas Lagrangian formalism provides a single set of second-order DEs. The Hamiltonian formulation possesses inherent advantages over the Lagrangian as a coupled set of first-order DEs is numerically more stable and more comfortable to solve than a single set of second-order DEs. Nevertheless, the resulting DEs are often analytically intractable, so engineers and scientists resort to discretization techniques to obtain solutions. However, the discretization procedure for solving the DEs could lead to violations of the underlying conservation laws. This issue can be cured by using NN solvers that able to provide analytical solutions that respect the underlying principles. Indeed, any sort of semi-implicit method, like symplectic Euler integrator, allows errors to accumulate in time. Chaotic systems, in particular, are highly sensitive to such concerns and are, therefore, an ideal ground for testing the performance of the proposed Hamiltonian NN.

We consider a physical system of many bodies that are moving in space. The motion of those objects can be described in a d -dimensional configuration space which is defined by the specification of the position as a function of the time t of all objects in a system. More precisely, d is defined as the product of the number of bodies in a system and the number of spatial dimensions that those objects are allowed to move. In the Lagrangian formulation we are working on the configuration space, whereas the Hamiltonian formalism is defined in the phase space, which consists of the position and momentum of the objects. Subsequently, each dimension in the configuration space associates with two degrees of freedom in the phase space. In this paper, we are interested in a Hamiltonian framework, therefore we consider a phase space of $D = 2d$ dimensions. Many classical systems, from the simple pendulum to solar systems, can be described by the separable Hamiltonian form $\mathcal{H} = T + V$, where the potential energy term V depends solely on the *generalized space coordinates* $\mathbf{q} = (q_1, \dots, q_d)$, and the kinetic term T depends solely on the *generalized momenta* $\mathbf{p} = (p_1, \dots, p_d)$. Since

this Hamiltonian form does not depend directly on time, systems described by it will conserve energy. Other dynamical invariants may also be inbuilt, depending upon the specific choice of the individual phase space variables and their corresponding continuous symmetries [43]. As an example, when the Hamiltonian does not directly depend on a coordinate q_i , the associated momentum p_i is conserved and vice versa. For such Hamiltonian functions, the dynamics are governed by following coupled DEs, called Hamilton's or *canonical* equations,

$$\dot{q}_i = \frac{\partial \mathcal{H}}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial \mathcal{H}}{\partial q_i}, \quad (1)$$

where dots denote time derivatives. An elegant way of expressing Hamilton's equations is the *symplectic* notation. Let $\mathbf{z} = (q_1, \dots, q_d, p_1, \dots, p_d)^T \in \mathbb{R}^D$ and \mathbf{J} be the $D \times D$ matrix

$$\mathbf{J} = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{1} & \mathbf{0} \end{pmatrix}, \quad (2)$$

where $\mathbf{0}$ and $\mathbf{1}$ represent the $d \times d$ zero and unity matrix, respectively. Then, Hamilton's equations can be written in the compact vector form

$$\dot{\mathbf{z}} = \mathbf{J} \cdot \nabla_{\mathbf{z}} \mathcal{H}(\mathbf{z}), \quad (3)$$

where $\nabla_{\mathbf{z}} \mathcal{H}(\mathbf{z}) = \partial \mathcal{H}(\mathbf{z}) / \partial \mathbf{z}$. Numerical methods that evaluate Eq. (3) are called symplectic methods and have been widely used to calculate the long-term evolution of chaotic systems [44]. In this paper, we present an alternative method based on NNs to solve Eq. (3). As we will discuss below, symplectic integrators conserve a Hamiltonian which is slightly perturbed from the original, whereas, symplectic NNs conserve the original Hamiltonian. This is a great advantage that the proposed NN has over the symplectic integrators.

An alternative approach to numerically solving DEs is offered by feed-forward NNs [18,20,24]. One key advantage of such NNs over traditional numerical methods is that they seek to learn actual functions that satisfy the DEs, rather than creating an approximation to the real solution. Moreover, the NN solutions are in a closed, differentiable, and analytic form [18], and the calculations can be efficiently implemented on parallel architectures leading to significant speedups [18]. The advantage in using our proposed NN architecture is that it provides solutions that satisfy Hamilton's equations simultaneously. Thus, the dynamical invariants of a particular Hamiltonian are being respected to the required precision, compared to the accumulation of errors that is inevitable in iterative solvers. To compare, we present the semi-implicit Euler method, which is the simplest, yet most widely used, symplectic integrator for solving Hamilton's equation. The symplectic Euler method conserves energy up to a fluctuating error because it conserves a slightly different Hamiltonian than the original. For the separable Hamiltonian form $\mathcal{H} = T(p_i) + V(q_i)$, the symplectic Euler scheme for solving the system (1) reads

$$q_i^{(n+1)} = q_i^{(n)} + \Delta t \frac{\partial T(p_i^{(n)})}{\partial p_i^{(n)}}, \quad (4)$$

$$p_i^{(n+1)} = p_i^{(n)} - \Delta t \frac{\partial V(q_i^{(n+1)})}{\partial q_i^{(n+1)}}. \quad (5)$$

Here, Δt is the time step between two sequential time points, (n) denotes the time point that is evaluated, $q_i^{(n)} = q_i(n\Delta t)$, and $p_i^{(n)} = p_i(n\Delta t)$. Due to the iterating nature of symplectic Euler method, we read in Eqs. (4) and (5) that the solutions at two sequential time points are needed to evaluate Hamilton's equations at any point, leading to numerical errors in the calculation of energy, that is, proportional to Δt . Similarly, higher-order iterative symplectic integrators accumulate numerical error, however, this paper presents a comparison only between the solutions obtained by the proposed NN solver and the first-order symplectic Euler integrator.

The objective of this study is to solve Hamilton's equation (3) in a certain time interval by using NNs. Let us consider the general form of parametric solutions,

$$\hat{\mathbf{z}}(t) = \mathbf{z}(0) + f(t)\mathbf{N}(t), \quad (6)$$

where $\hat{\mathbf{z}}$ is the solution vector discovered by the NN, $\mathbf{z}(0)$ is the initial state vector, and $\mathbf{N}(t) \in \mathbb{R}^D$ is a vector of D outputs of a feed-forward fully connected NN. The parametric function $f(t)$ enforces the initial conditions in the parametric solutions, i.e., $\hat{\mathbf{z}}(0) = \mathbf{z}(0)$ when $f(0) = 0$. The network takes as a single input the time point t_n , where n denotes the n th sequential point; without losing the generality, we consider the initial time $t_0 = 0$. We train the NN by minimizing, with respect to the learning parameters of the network, the mean-squared error (MSE) defined by Hamilton's equation (3) as

$$L = \frac{1}{K} \sum_{n=1}^K (\dot{\hat{\mathbf{z}}}^{(n)} - \mathbf{J} \cdot \nabla_{\hat{\mathbf{z}}^{(n)}} \mathcal{H}(\hat{\mathbf{z}}^{(n)}))^2 + \lambda L_{\text{reg}}, \quad (7)$$

where $\hat{\mathbf{z}}^{(n)} = \hat{\mathbf{z}}(t_n)$ and K is the total number of the input time points used for the network optimization. The term L_{reg} can be any regularization term where λ is the regularization parameter. We have found that for long-time predictions it is efficient to use a regularization term that penalizes violations of the energy conservation law. Given the initial state and corresponding energy, E_0 , of a system, a convenient regularization term is

$$L_{\text{reg}} = \frac{1}{K} \sum_{n=1}^K [(\mathcal{H}(\hat{\mathbf{z}}^{(n)}) - E_0)^2]. \quad (8)$$

For long-time prediction, the use of the regularization loss (8) stabilizes the predicted trajectory at the correct energy level and can result in faster network convergence. In the present paper, results are presented for $\lambda = 0$ unless otherwise specified.

The time derivatives are obtained by using automatic differentiation that computationally costs one back-propagation through the entire network [41]. We first generate equally spaced time points t_n in the training time interval $[0, T]$. Then, we randomly perturb these points in each epoch as: $t_n \rightarrow t_n + \epsilon$ where ϵ is a random term obtained by a normal distribution [20]. This trick improves the network predictability as it is effectively trained over a continuous time interval. In addition, perturbing the training points in every epoch employs the stochastic gradient descent (SGD) method, and thus it assists the optimizer to escape from local minima in the loss function. Perturbing the points in every epoch means that we perturb the loss function and, subsequently, the local

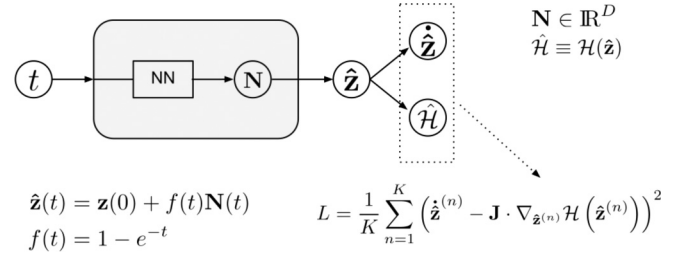


FIG. 1. Hamiltonian architecture with parametrization $\hat{\mathbf{z}}(t)$ used in the loss function L ; \mathcal{H} is the Hamiltonian and $f(t)$ imposes the initial conditions to $\hat{\mathbf{z}}(t)$; K is the number of the training points and (n) indicates each time point.

minima are dynamically moving. In the context of SGD, each epoch is considered as a minibatch while all the epochs consist the whole batch for the training set. Minimizing the loss function in Eq. (7) yields solutions that identically respect the symplectic structure of Eq. (3) and, accordingly, every dynamical invariant of the Hamiltonian flow is respected too. We point out that the NN solutions are of high accuracy when the NN is evaluated in the training interval $[0, T]$ but the error rapidly increases outside of the training interval. The proposed Hamiltonian NN architecture is graphically demonstrated in Fig. 1. It is worth noting that the proposed network of Fig. 1 has a different architecture than one used in standard Hamiltonian NNs [32]. Our network takes t_n as input and returns $\hat{\mathbf{z}}^{(n)}$, whereas the input in the standard Hamiltonian networks is $\mathbf{z}^{(n)}$ and the output is $\hat{\mathbf{z}}^{(n+1)}$.

A crucial role in the performance of the NN is played by $f(t)$. A standard choice to enforce initial conditions is $f(t) = t$, which satisfies $f(0) = 0$ [18]. However, this is an unbounded function that adds further difficulty when t becomes large. Specifically, for the NN outputs after enough epochs, Eq. (6) states that $\mathbf{N} = (\hat{\mathbf{z}} - \mathbf{z}(0))/t$. As t increases, the \mathbf{N} tends to zero, which negatively affects the network predictability in large timescales. To remedy this inefficiency, we propose the parametric function

$$f(t) = 1 - e^{-t}, \quad (9)$$

which is a smooth, bounded function, with $f(0) = 0$. Later, we show that the specific choice of parametric function drastically improves the predictability of the NN solver. Interestingly enough, the fact that $f(t)$ rapidly tends to 1 implies that the proposed architecture consists of a symplectic NN. In particular, for $\lambda = 0$ and at the limit $L \rightarrow 0$, Eq. (7) yields $\dot{\hat{\mathbf{z}}} = \mathbf{J} \cdot \nabla_{\hat{\mathbf{z}}} \mathcal{H}(\hat{\mathbf{z}})$, and as $t \rightarrow \infty$, we have $\hat{\mathbf{z}} = \mathbf{z}(0) + \mathbf{N}$. Considering the aforementioned two limits and performing the linear transformation $\mathbf{N} \rightarrow \mathbf{N} - \mathbf{z}(0)$, we obtain

$$\dot{\mathbf{N}} = \mathbf{J} \cdot \nabla_{\mathbf{N}} \mathcal{H}(\mathbf{N}), \quad (10)$$

which indicates that the proposed architecture comprises a symplectic NN that states that the function $\mathcal{H}(\mathbf{N})$ is time invariant.

A substantial advance that our method suggests is the energy regularization of Eq. (8). Because of the absence of iteration learning, the NN solver does not build the solutions using predictions from previous steps. As a result, it tends to forget the initial state of the system and thus, in long-

time solutions, energy leaking might be observed resulting in error accumulation. This issue becomes crucial in long-time solutions reducing the ability of solving nonlinear and especially chaotic systems of ODEs. The regularization loss of Eq. (8) stabilizes the predicted trajectories at the correct energy yielding a robust solver. Another important innovation of this paper is the choice of the activation function. It has been shown that NN with trigonometric activation functions can learn periodic behavior from data outperforming networks that use common activations like Relu and Sigmoid [45]. We adopt this approach and choose the trigonometric $\sin(\cdot)$ as the activation function. Empirical results presented later through numerical experiments indicate that $\sin(\cdot)$ activation outperforms sigmoid in solving ODEs for Hamiltonian systems.

B. Error analysis

We seek to provide a rough bound on the error in the solution based on the maximum value of the loss function. To begin, note that Eq. (7) can be written as

$$L = \frac{1}{K} \sum_{n=1}^K (\ell_n^2 + \lambda \ell_{\text{reg},n}^2), \quad (11)$$

where

$$\ell_n = \hat{\mathbf{z}}^{(n)} - \mathbf{J} \cdot \nabla_{\mathbf{z}^{(n)}} \mathcal{H}(\hat{\mathbf{z}}^{(n)}), \quad (12)$$

$$\ell_{\text{reg},n} = \mathcal{H}(\hat{\mathbf{z}}^{(n)}) - E_0, \quad (13)$$

and $\ell_n, \ell_{\text{reg},n} \in \mathbb{R}^K$ are vectors containing the respective loss components at some arbitrary time point t_n . Since L is the loss function for the NN, averaged over time, ℓ_n^2 can be considered the instantaneous loss at the n th time point when $\lambda = 0$. Let $\delta \mathbf{z} = \mathbf{z} - \hat{\mathbf{z}}$ be the error between the true solution and the NN solution. Expanding the Hamiltonian $\mathcal{H}(\mathbf{z}) = \mathcal{H}(\hat{\mathbf{z}} + \delta \mathbf{z})$ in a Taylor series about $\hat{\mathbf{z}}$ and keeping up to quadratic terms yields

$$\mathcal{H}(\mathbf{z}) \approx \mathcal{H}(\hat{\mathbf{z}}) + (\nabla_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z} + \frac{1}{2} (\mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}^2, \quad (14)$$

where $\mathcal{D}_{\mathbf{z}}$ is the Hessian matrix. Taking the gradient of Eq. (14) with respect to \mathbf{z} and rearranging terms gives

$$(\nabla_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \approx \nabla_{\mathbf{z}} \mathcal{H}(\mathbf{z}) - (\mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}. \quad (15)$$

We note that for Hamiltonians with quadratic dependence on \mathbf{z} , the quadratic expansion (14) is exact because higher order terms vanish. In addition, the second order in $\delta \mathbf{z}$ is the smallest order still large enough to not be canceled when we move to substitute Eq. (15) into Eq. (12). Nevertheless, the derivation can be extended to include higher order terms in a straightforward manner. In what follows, we drop the superscript (n) for clarity of presentation. Substituting the Taylor series expansion (15) into (12) and invoking (3) results in

$$\ell \approx \mathbf{J} \cdot [(\mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}] - \delta \mathbf{z}. \quad (16)$$

Inspecting the vector DE (16), we observe that its components comprise a closed differential system for the error δz_i in each predicted trajectory \hat{z}_i . Solving this differential system with initial condition $\delta \mathbf{z}(0) = 0$, as dictated by the parametrization (6), we can compute how the errors propagate in time. However, this requires knowledge of the loss components of $\ell(t)$ and thus, such an analysis can be performed only after we have trained the network.

On the other hand, we can derive a bound on the size of $\delta \mathbf{z}$ without having exact knowledge of $\ell(t)$ by constructing a worst case scenario. We want to establish a relationship between ℓ and $\delta \mathbf{z}$, such that it determines when to stop the network training to get solutions with better than a certain accuracy. Let $\ell_{\text{max}}^2 = \max_t (\ell^2 + \lambda \ell_{\text{reg}}^2)$ represent the largest instantaneous loss that the NN will have after being trained. In the following analysis, we denote the 2– norm by $\|\cdot\|$. We have

$$\begin{aligned} \ell_{\text{max}}^2 &\geq \|\ell\|^2 + \lambda \|\ell_{\text{reg}}\|^2 \geq \|\ell\|^2 \\ &= \|\dot{\delta \mathbf{z}} - \mathbf{J} \cdot (\mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}\|^2 \\ &\geq \|\dot{\delta \mathbf{z}}\| - \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}\|^2 \\ &= \|\dot{\delta \mathbf{z}}\|^2 - 2\|\dot{\delta \mathbf{z}}\| \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}\| \\ &\quad + \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}\|^2 \\ &\geq \|\dot{\delta \mathbf{z}}\|^2 - 2\|\dot{\delta \mathbf{z}}\| \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}\| + (\sigma_{\min} \|\delta \mathbf{z}\|)^2, \end{aligned} \quad (17)$$

where σ_{\min} is the minimum singular value of $(\mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}}$. The last line in the above expression (17) can be obtained by considering the quantity $\|A\mathbf{x}\|$ and using the singular value decomposition on A to show that $\|A\mathbf{x}\| \geq \sigma_{\min} \|\mathbf{x}\|$. Rearranging terms leads to

$$\begin{aligned} \sigma_{\min}^2 \|\delta \mathbf{z}\|^2 &\leq \ell_{\text{max}}^2 - \|\dot{\delta \mathbf{z}}\|^2 + 2\|\dot{\delta \mathbf{z}}\| \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}} \delta \mathbf{z}\| \\ &\leq \ell_{\text{max}}^2 - \|\dot{\delta \mathbf{z}}\|^2 + 2\|\dot{\delta \mathbf{z}}\| \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}}\| \|\delta \mathbf{z}\| \\ &\Rightarrow \sigma_{\min}^2 \|\delta \mathbf{z}\|^2 - 2\|\dot{\delta \mathbf{z}}\| \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}}\| \\ &\quad \times \|\delta \mathbf{z}\| \leq \ell_{\text{max}}^2 - \|\dot{\delta \mathbf{z}}\|^2. \end{aligned} \quad (18)$$

Solving the quadratic inequality (18) for $\|\delta \mathbf{z}\|$ yields

$$\begin{aligned} \|\delta \mathbf{z}\| &\leq \frac{\|\dot{\delta \mathbf{z}}\| \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}}\|}{\sigma_{\min}^2} \\ &\quad + \frac{1}{\sigma_{\min}^2} [\sigma_{\min}^2 \ell_{\text{max}}^2 - \|\dot{\delta \mathbf{z}}\|^2 \\ &\quad \times (\sigma_{\min}^2 - \|(\mathbf{J} \cdot \mathcal{D}_{\mathbf{z}} \mathcal{H}(\mathbf{z}))_{\hat{\mathbf{z}}}\|^2)]^{1/2}. \end{aligned} \quad (19)$$

Now consider a single component of the error, δz_i . The largest value δz_i can take occurs when $\delta z_i \neq 0$ and $\delta z_j = 0$ for $j \neq i$. That is, for a fixed error, all of the error is concentrated in a single component. In this case, $\|\delta \mathbf{z}\|^2 = \delta z_i^2$. If δz_i^2 is maximized at a value t_{max} , then $(\dot{\delta z}_i^2) = 0$ at t_{max} . Therefore, $\delta z_i \dot{\delta z}_i = 0 \Rightarrow \delta z_i = 0$. Using this in (19) provides

$$\|\delta z_i\| \leq \frac{\ell_{\text{max}}}{\sigma_{\min}}. \quad (20)$$

We point out that at the boundary $t = 0$, the error and its derivatives are exactly zero since the initial conditions are identically satisfied through the parametrization of Eq. (6). Furthermore, the assumption that all the error is concentrated at a single component z_i implies that δz_j and $\dot{\delta z}_j$ are zero functions for $j \neq i$. This strong assumption simplifies Eq. (19) yielding the upper error bound of Eq. (20).

If a NN is trained such that the loss function has a maximum value of ℓ_{max} , then the maximum error that any component of the solution can take is bounded by (20). In other words, we can choose in advance an accuracy for the solutions and use the relationship (20) to calculate the ℓ_{max} ,

which, therefore, will determine when we have to stop training the network ensuring the desirable accuracy. The σ_{\min} can be calculated due to the training process since, in the most general case, it is a function of the solutions. Moreover, the expressions (16) and (20) state that $|\delta \mathbf{z}|$ depends on the general network performance and not only on the number of time points used in the training process, which is the case of numerical integrators. That happens because the number of training points is not the only parameter that determines the value of the loss function. For example, fixing the number of the training points while increasing the number of hidden layers or neurons yields better performance that corresponds to a smaller ℓ_{\max} . In summary, once the Hamiltonian NN is optimized, Eq. (16) can be used to calculate the error propagation. On the other hand, we can decide the accuracy of the solutions before the optimization by using Eq. (20) to define the ℓ_{\max} that determines when to stop training the network.

III. EXPERIMENTS

A. Nonlinear oscillator

As a concrete example, we consider the one dimensional nonlinear (anharmonic) oscillator with Hamiltonian

$$\mathcal{H} = \frac{p^2}{2} + \frac{x^2}{2} + \frac{x^4}{4}, \quad (21)$$

where the natural frequency and the mass of the oscillator are considered to be unity. The Hamiltonian (21) corresponds to the total energy E of the system, and the associated equations of motion read [Eqs. (1)]

$$\dot{x} = p, \quad \dot{p} = -(x + x^3). \quad (22)$$

In what follows, we use the symplectic NN architecture to solve the above nonlinear Hamiltonian system and compare the NN solutions with those obtained by symplectic Euler integrator. It results that the symplectic Euler method requires two orders more evaluation time points than the NN to reach the same numerical error. We also explore the efficiency of the network for different activation and parametric functions.

The phase space of the oscillator consists of two degrees of freedom with $\mathbf{z} = (x, p)^T$. Accordingly, we utilize a feed-forward NN with two outputs $\mathbf{N} = (N_1, N_2)^T$ used to parametrize the approximate solutions $\hat{\mathbf{z}} = (\hat{x}, \hat{p})^T$ according to Eq. (6). The loss function is defined by Eqs. (22) and according to Eq. (7) as

$$L = \frac{1}{K} \sum_{n=1}^K [(\dot{\hat{x}}^{(n)} - \dot{\hat{p}}^{(n)})^2 + (\dot{\hat{p}}^{(n)} + \hat{x}^{(n)} + (\hat{x}^{(n)})^3)^2]. \quad (23)$$

We initialize a grid with $K = 200$ time points equally spaced in the time interval $t = [0, 4\pi]$. At the beginning of each epoch, we perturb all the time points by using a random term obtained by a normal distribution with zero mean and a standard deviation of 0.06π . The initial state is chosen to be $(x_0, p_0) = (1.3, 1.0)$, corresponding to the total initial energy $E_0 = 2.06$; in this energy, the motion deviates from the behavior of the simple harmonic oscillator. The NN consists of two hidden layers with 50 neurons per hidden layer, and is being trained for 5×10^4 epochs by using Adam optimizer [46] with a learning rate of 8×10^{-3} . We perform

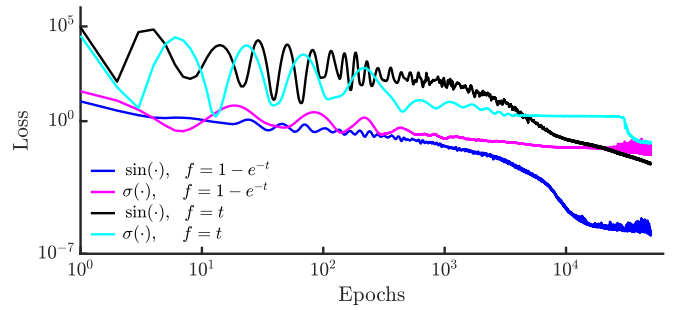


FIG. 2. Hamiltonian NN solves the equations of the nonlinear oscillator system. Color lines represent the loss function in log-scale during the training for different combinations of activation and parametric functions f shown in legend.

four independent numerical experiments that correspond to different NN designs, namely, for the combinations of sigmoid $\sigma(\cdot)$ and trigonometric $\sin(\cdot)$ activation functions, and for the parametric functions $f(t) = t$ and $f(t) = 1 - e^{-t}$. Figure 2 demonstrates in logarithmic scale the loss function (23) during the training; each color represents one of the the four distinguished cases of architectures according to the legend. We highlight that the loss function of our proposed design (blue line) converges faster than the other models.

The performance of the Hamiltonian NN after its training is represented in Fig. 3 by the blue curve. In addition, we use the DEs solver ODEINT of the SCIPY python package [47] to solve system (22) and consider the obtained numerical solutions as the ground truth. We note that the solvers provided by `scipy` have exemplary error control and adaptivity leading to excellent solution trajectories. For comparison purposes, we also utilize the symplectic Euler method described in Eqs. (4) and (5) to solve the DEs (22), and compare the solutions with those obtained by our proposed symplectic NN. We point out that the ground truth data and the solutions obtained by symplectic Euler method are exclusively used to assess the performance of the NN predictions and never used for the NN optimization. Essentially, the Hamiltonian NN does not use any data generated by traditional numerical solvers. In Fig. 3, we present results obtained by the solver (green lines), by the NN (blue line), and by the symplectic Euler integrator (black and red). After the network optimization, we get $\ell_{\max} = 3.3 \cdot 10^{-3}$. The smallest singular value of the Hessian of Hamiltonian (21) is $\sigma_{\min} = 1$. Subsequently, Eq. (20) yields for both δx and δp the upper bound error 3.3×10^{-3} . Interestingly enough, the symplectic Euler method needs $100 \times K$ time points to approach this maximum error. In the case of Euler's method, we present in Fig. 3 two numerical solutions: one with the same time points K used in the NN training (black) and a second with 100 times more points (red). The left graph in Fig. 3 demonstrates the phase space for the numerical errors where we observe that the errors in the NN's solutions are in the same order with the error obtained by the symplectic Euler when 100 times more time points are used. In the right panel of Fig. 3, we present $\delta x(t)$ and $\delta p(t)$ and the the total energy as a function of time calculated by using the numerical solutions in the Hamiltonian (21). An important result of this exploration is that, in contrast to the Euler integrator, the NN's solutions conserve the total energy

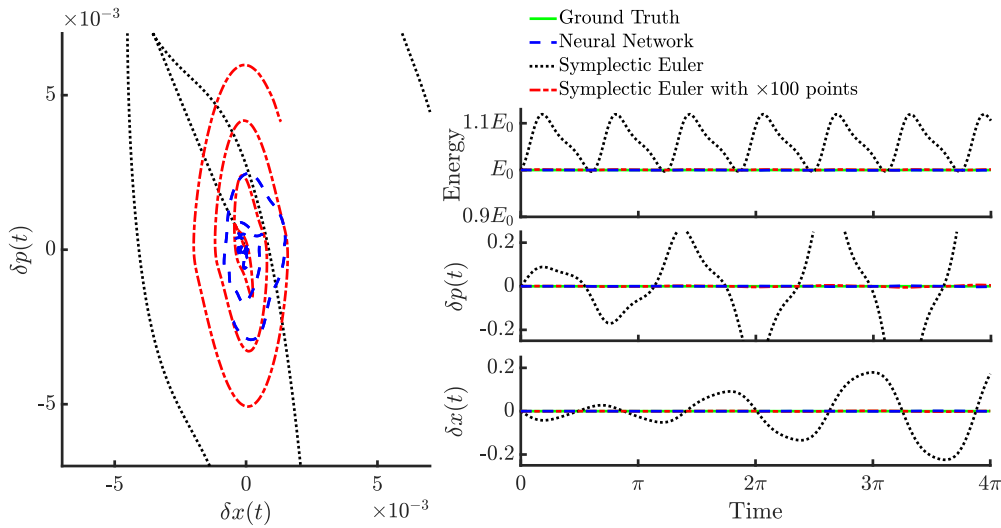


FIG. 3. Comparing the ground truth (green) with the approximated solutions obtained by NN (blue) and by symplectic Euler integrator. The NN is trained over $K = 200$ time points while the integrator is evaluated at K (black) and $100 \times K$ (red) points. Left: The phase space of the numerical error. Right: The error evolution in position and momenta, and the total energy in time.

locally. This is a consequence of the fact that the solutions obtained by the symplectic NN conserve the correct Hamiltonian rather than a perturbed one, which is the case with the symplectic integrators. Therefore, in the context of the energy conservation task, the Hamiltonian NN outperforms the symplectic Euler integrator.

We validate the predictability of the Hamiltonian NN for long-term prediction by solving DEs in a longer time period. In particular, the system of Eqs. (22) is solved for the extended time interval $[0, 20\pi]$ using the same initial conditions from the previous simulations, namely $(x_0, p_0) = (1.3, 1.0)$. Although the previously used architecture provides solutions of high accuracy, we found that by using 80 neurons per hidden layer yields faster convergence in the NN optimization. Moreover, since the time interval is expanded, the number of training points is increased accordingly to $K = 500$ time points. For the long-time prediction in this case, we use the regularization loss (7) with $\lambda = 1$, which penalizes violations in the energy conservation. The NN predictions are compared

to solutions obtained by the symplectic Euler method using 5×10^4 points. This represents 100 times more points than those used for the network optimization. The results of the long-time solutions are presented in Fig. 4. The loss function during the training is shown by the left upper image in Fig. 4. The lower panel represents the ground truth energy (green solid line), the energy obtained by the Hamiltonian NN (dashed blue), and the energy that the symplectic Euler method (red dashed-dotted) computes. We observe that the NN conserves the energy slightly better than the numerical integrator. The right panel of Fig. 4 is the phase-space error, similar to Fig. 3, where we observe that the error obtained by the symplectic Euler (red dashed-dotted) constantly increases in time much faster than the error that we obtain from the NN solutions. Interestingly enough, we observe that although the energy is conserved comparably well by the two approaches, the Hamiltonian NN outperforms the symplectic Euler method in terms of the accuracy of the predicted solutions. That happens because a NN solver simultaneously

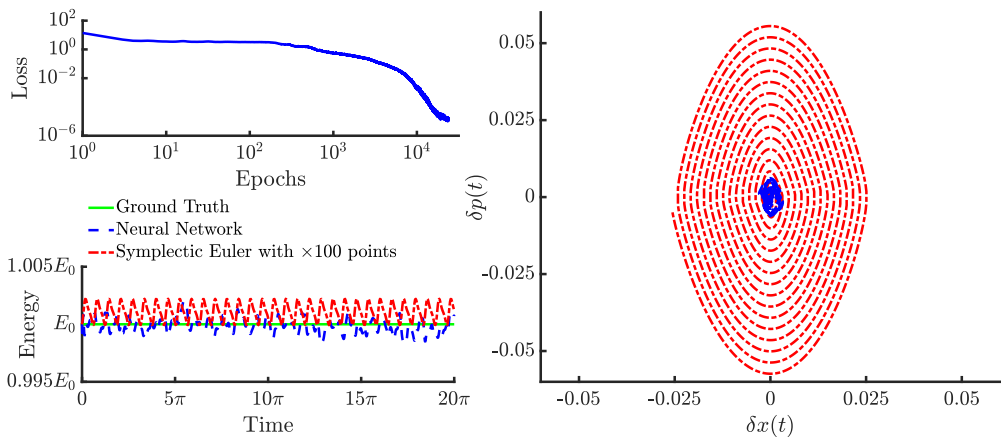


FIG. 4. Long-term prediction. The upper left image demonstrates the loss function during the training of a Hamiltonian network. The lower left graph shows the ground truth energy along with the predictions obtained by NN and by symplectic Euler method. The right panel presents the phase space of the numerical error for the predicted solutions.

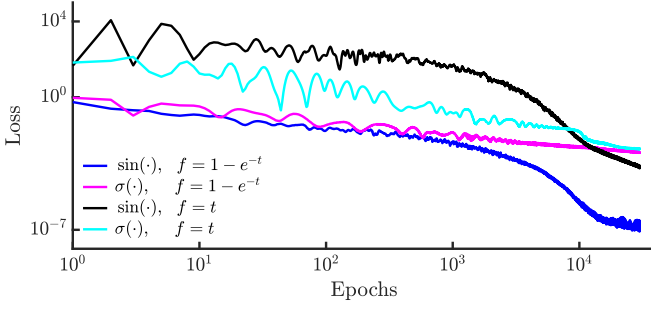


FIG. 5. NN solves the equations of motion for the HH system. Loss function in log scale during the training for a different combinations of activation and parametric functions f shown by the legend.

satisfies all the equations of DE system and conserves the original Hamiltonian, whereas, integrators conserve a perturbed Hamiltonian accumulating errors in time.

B. Chaotic system

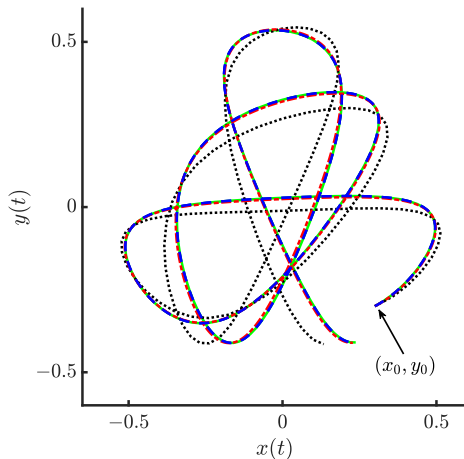
We demonstrate further the efficiency of the proposed symplectic NN by solving the equations for a chaotic two-dimensional dynamical system. In particular, we solve the canonical equations for the Hénon-Heiles (HH) system [48] that describes the nonlinear motion of a star around a galactic center with the motion restricted to a plane. The HH system has four degrees of freedom in the phase space where $\mathbf{z} = (\mathbf{q}, \mathbf{p})^T = (x, y, p_x, p_y)^T$. The Hamiltonian and the total energy of this system is

$$\mathcal{H} = \frac{1}{2}(p_x^2 + p_y^2) + \frac{1}{2}(x^2 + y^2) + \left(x^2y - \frac{y^3}{3}\right). \quad (24)$$

The Hamilton's equations results in the nonlinear DE system:

$$\dot{x} = p_x, \quad \dot{y} = p_y, \quad (25)$$

$$\dot{p}_x = -(x + 2xy), \quad \dot{p}_y = -(y + x^2 - y^2). \quad (26)$$



For the HH system, we are seeking approximate solutions $\hat{\mathbf{z}} \in \mathbb{R}^4$. Accordingly, we employ a fully connected feed-forward NN with four outputs $\mathbf{N} \in \mathbb{R}^4$ used to parametrize $\hat{\mathbf{z}}$ according to the general formula (6). The initial conditions for the numerical experiment are $(x_0, y_0, p_{x,0}, p_{y,0}) = (0.3, -0.3, 0.3, 0.15)$, corresponding to the energy $E_0 = 0.13$. The maximal Lyapunov exponent for this set of initial conditions is $\nu = 0.069$, and since ν is positive, the motion is chaotic [49]. The network consists of two hidden layers with 50 neurons per hidden layer. An equally spaced grid of $K = 100$ is initialized in the time interval $t = [0, 6\pi]$ that corresponds to 1.3 Lyapunov times. These points are used as the training set and are perturbed in the beginning of every epoch by using a random term obtained by a normal distribution with zero mean and with a standard deviation 0.18π . The loss function is defined by Eqs. (25) and (26), and according to Eq. (7), as

$$L = \frac{1}{K} \sum_{n=0}^K [(\hat{x}^{(n)} - \hat{p}_x^{(n)})^2 + (\hat{y}^{(n)} - \hat{p}_y^{(n)})^2 + (\hat{p}_x^{(n)} + \hat{x}^{(n)} + 2\hat{x}^{(n)}\hat{y}^{(n)})^2 + (\hat{p}_y^{(n)} + \hat{y}^{(n)} + (\hat{x}^{(n)})^2 - (\hat{y}^{(n)})^2)^2]. \quad (27)$$

We examine four different network architectures similar to the nonlinear oscillator system, namely for different activation and parametric functions. The networks are trained for 3×10^4 epochs by using Adam optimizer with learning rate 8×10^{-3} . After training for long enough to ensure convergence in the loss function, we find this number of epochs is sufficient to optimize the network. In Fig. 5, we show the loss function (27) in the training where each color corresponds to a different architectures according to the legend in the figure. Again, the choice of $\sin(\cdot)$ activation and $f(t) = 1 - e^{-t}$ yields the best network performance. In Fig. 6, we compare the approximated trajectories and the energy obtained by the symplectic NN (blue lines), and by a symplectic Euler integrator which is evaluated in K and in $10 \times K$ time points (shown

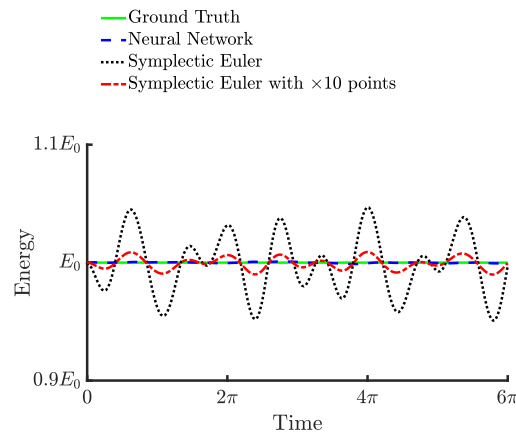


FIG. 6. Left: The orbit for the HH system in the x - y plane obtained by a NN (blue) that is trained in $K = 100$ time points and by symplectic Euler integrator evaluated in K (green) and $10 \times K$ (orange) points. Red curves are considered as the ground truth and obtained by a numerical solver. Right: Energy of the HH system with time. The Hamiltonian NN conserves energy locally while the symplectic Euler method does not maintain constant energy levels even at the highest resolution.

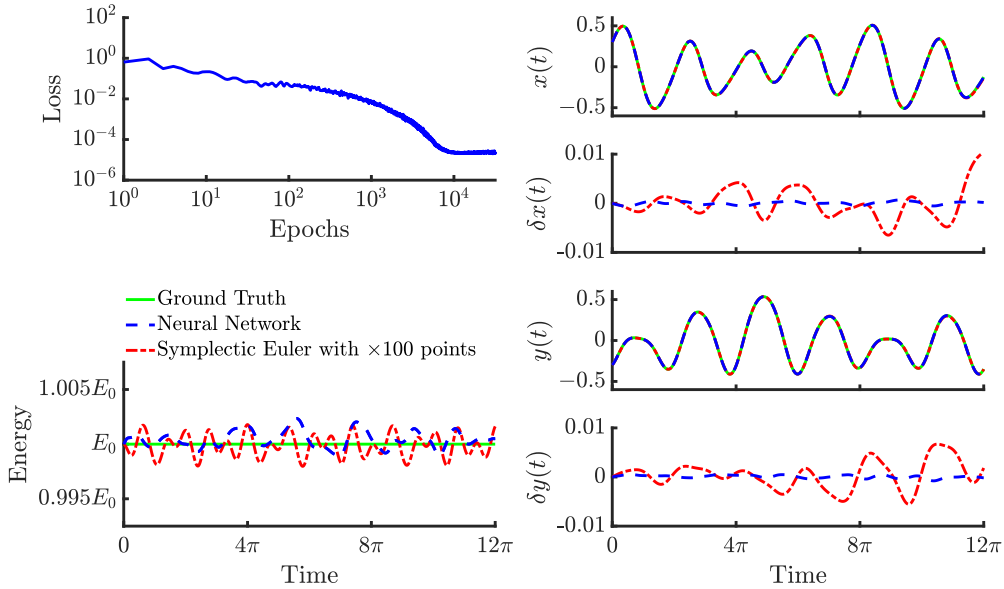


FIG. 7. Prediction on 2.6 Lyapunov times for the HH system. The left upper graph indicates the training loss of a Hamiltonian network. The left lower plot is the total energy obtained by Hamiltonian NN and by symplectic Euler evaluated in $10\times$ more than the training points. The right panels represent the predicted position states $x(t)$, $y(t)$ along with the associate numerical errors $\delta x(t)$ and $\delta y(t)$.

by black and red lines, respectively). Solutions obtained by a solver are considered as the ground truth (green curves). The left panel in Fig. 6 shows the orbit in the $x - y$ plane where the Hamiltonian NN solution is indistinguishable from the ground truth. The right panel represents the total energy in time where the NN solutions conserve the energy better than the solutions obtained by the symplectic Euler method. The symplectic Euler must use an order of magnitude higher resolution than NN to capture the correct orbit portrait, however, the energy is still not conserved locally.

We extend the integration time for the HH system to $[0, 12\pi]$, which corresponds to 2.6 Lyapunov times. For the long-time prediction, we employ the regularization term L_{reg} of Eq. (7) with $\lambda = 0.5$. The network architecture consists of two hidden layers with 80 neurons per layer. The network optimization uses 500 time points. For the training of this model, we found that using sequential learning [29] is more efficient. First, we train the model for a short integration time range of $[0, 6\pi]$ and save the network parameters; the network is trained for 2×10^4 epochs with a learning rate of 8×10^{-3} . Then, we load the previously saved parameters and train the model in a larger domain of $[0, 12\pi]$ for 5×10^4 epochs and with learning rate of 5×10^{-3} . This transfer learning application enhances the learning and, therefore, the network converges faster to the solutions than starting the training from random initialized parameters. The results are demonstrated by Fig. 7 where the left upper graph indicates the loss function during the training of the Hamiltonian network. For comparison, we present the NN results in blue along with the solutions obtained by a symplectic Euler evaluated with $10\times$ more points than the training points. The lower plot in left panel shows the energy where we observe that both the NN (blue) and the symplectic Euler (red) conserve the correct (green) energy with an error of about the same order. The right panel of Fig. 7 represents the predictions of the position state $x(t)$ and $y(t)$ along with the associated numerical error denoted

by δx and δy , respectively. As we observed in the nonlinear oscillator system, the solutions obtained by the NN presents a lower numerical error than the symplectic integrator, although both methods conserve the energy comparably well.

IV. CONCLUSION

In recent years, machine learning has made inroads in traditional science and engineering fields. NNs have attracted scientists' interest due to their outstanding capabilities in regression, classification, and prediction tasks. Since these methods are relatively new to physics, there are many physical concepts that have not been embedded yet in the structure of NNs. In this paper, we proposed a physics-inspired unsupervised NN for solving DEs that describe the temporal motion of dynamical systems. The Hamiltonian formulation is embedded in the NN through the loss function and, therefore, the predicted solutions conserve energy. The loss function is solely constructed by the network predictions and does not use any ground truth data. The proposed method does not use any data generated by traditional numerical solvers. Hence, the proposed Hamiltonian network provides a data-free unsupervised learning method. Although the Hamiltonian network presented in the current paper is an unsupervised model, the generalizations to the proposed network could incorporate data in a semisupervised fashion. Nevertheless, in this paper we focused on the exploration of the baseline unsupervised model and leave the semisupervised case for future work.

A smooth and bounded parametric form of solutions was introduced in this paper that makes the proposed architecture a symplectic network and, subsequently, a time-invariant unit. By appropriately choosing the activation function, a better domain knowledge is provided that drastically improves the network performance. Moreover, the proposed Hamiltonian architecture allows the network outputs to share their weights. Sharing the learning parameters helps the NN to discover

underlying code dependencies and subsequently improves the network predictability in learning solutions that satisfy nonlinear systems of DEs. The Hamiltonian structure of the proposed NN allows the use of a regularization term that penalizes violation in the energy conservation law. This penalty drastically improves the network performance especially for long-time solutions. The experiments presented in this paper indicate that to get accurate solutions for larger integration times, more hidden neurons and time points are required, increasing the network complexity and the computational cost. This cost can be potentially reduced by parallelizing the calculations since each time point is treated independently, however, such an implementation is not presented in this paper. In the limit of very long integration times, we expect to need very large network complexity and batches of time points, thus, a parallel implementation will be crucial. An error analysis was developed in this paper which can be used to analyze how the errors in the predicted solutions propagate in time. In addition, this error analysis provides a threshold in the loss function, where we can early-stop training the network when a certain accuracy occurs, namely, a lower error in the predicted solutions is ensured.

There are several advantages in using NN solvers instead of traditional symplectic numerical integrators for solving DEs. The solutions obtained by a NN are continuous, smooth, and in analytical form. Due to many outputs with shareable weights, the Hamiltonian NN discovers solutions that satisfy the Hamilton equations simultaneously and consistently. Subsequently, the NN solver conserves the correct Hamiltonian in contrary to symplectic integrators that conserve a slightly perturbed Hamiltonian. We outlined that the solutions obtained by the NN conserve the energy locally along with all the time points and outperforms the symplectic Euler integrator that predicts an energy with a fluctuating error term. In addition to the first-order Euler method, there are higher order symplectic integrators that accumulate less error than a semi-implicit Euler but with a larger computational cost. Such a comparison between the proposed NN solver and higher

order integrators is not presented in this paper. In problems where energy conservation is crucial, the Hamiltonian NN will show better performance than symplectic integrators. Moreover, NN solvers can potentially possess advantages over state-of-the-art integrators such as the ODEINT from the SCIPY Python package. As pointed out by Ref. [18], the calculations for a NN can be efficiently implemented on parallel architectures, leading to significant speedup. This is possible because NN solvers evaluate the time points independently. In the years since that original work of Ref. [18] appeared, hardware innovations such as GPUs have made the parallelization of NNs even more accessible. On the contrary, time-parallel algorithms for traditional numerical integrators are challenging to develop and implement since the computation at a time point requires solutions at prior time points. Additionally, as the number of DEs in a system increases the problem of the curse of dimensionality is observed, making the numerical integrators inefficient due to the rapidly increase of the computational cost. On the other hand, it has been shown by Refs. [19,20] that the problem of the curse of dimensionality does not occur in NN DE solvers. Subsequently, in high-dimensional problems such as many-body problems, we expect the Hamiltonian NNs to outperform regular symplectic integrators. Considering that Hamiltonian formulation provides a solid framework for theoretical extension in many areas of physics such as perturbation approaches and theory of chaos, as well as statistical and quantum mechanics, the proposed Hamiltonian NN provides fertile ground on which modern research problems can potentially be handled.

ACKNOWLEDGMENTS

The authors would like to thank Prof. E. Lagaris, Prof. G. P. Tsironis, and Prof. E. Kaxiras for fruitful discussions. A.S.D. acknowledges support by the President's Scholarship at Imperial College London and by the EPSRC Centre for Doctoral Training in Mathematics of Random Systems: Analysis, Modelling and Simulation (EP/S023925/1).

-
- [1] B. Lusch, J. N. Kutz, and S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, *Nat. Commun.* **9**, 4950 (2018).
 - [2] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks, *Proc. R. Soc. A: Math. Phys. Eng. Sci.* **474**, 20170844 (2018).
 - [3] G. Neofotistos, M. Mattheakis, G. D. Barnmparis, J. Hizanidis, G. P. Tsironis, and E. Kaxiras, Machine learning with observers predicts complex spatiotemporal behavior, *Front. Phys.* **7**, 24 (2019).
 - [4] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, Reservoir observers: Model-free inference of unmeasured variables in chaotic systems, *Chaos* **27**, 041102 (2017).
 - [5] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Model-Free Prediction of Large Spatiotemporally Chaotic Systems From Data: A Reservoir Computing Approach, *Phys. Rev. Lett.* **120**, 024102 (2018).
 - [6] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: A review, *Neural Networks* **115**, 100 (2019).
 - [7] F. Karim, S. Majumdar, H. Darabi, and S. Harford, Multivariate LSTM-FCNs for time series classification, *Neural Networks* **116**, 237 (2019).
 - [8] J. Ling, R. Jones, and J. Templeton, Machine learning strategies for systems with invariance properties, *J. Comput. Phys.* **318**, 22 (2016).
 - [9] J. Ling, A. Kurzawski, and J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, *J. Fluid Mech.* **807**, 155 (2016).
 - [10] R. Fang, D. Sondak, P. Protopapas, and S. Succi, Neural network models for the anisotropic Reynolds stress tensor in turbulent channel flow, *J. Turbul.* **21**, 525 (2020).
 - [11] K. Duraisamy, G. Iaccarino, and H. Xiao, Turbulence modeling in the age of data, *Annu. Rev. Fluid Mech.* **51**, 357 (2019).
 - [12] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, *J. Comput. Phys.* **335**, 736 (2017).

- [13] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, *J. Comput. Phys.* **348**, 683 (2017).
- [14] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, Data-driven discovery of partial differential equations, *Sci. Adv.* **3** (2017).
- [15] J. N. Kutz, S. H. Rudy, A. Alla, and S. L. Brunton, Data-driven discovery of governing physical laws and their parametric dependencies in engineering, physics and biology, *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)* (IEEE, Curacao, 2017), pp. 1–5.
- [16] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner, Learning data-driven discretizations for partial differential equations, *Proc. Natl. Acad. Sci. USA* **116**, 15344 (2019).
- [17] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* **378**, 686 (2019).
- [18] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Netw.* **9**, 987 (1998).
- [19] J. Han, A. Jentzen, and E. Weinan, Solving high-dimensional partial differential equations using deep learning, *Proc. Natl. Acad. Sci. USA* **115**, 8505 (2018).
- [20] J. A. Sirignano and K. Spiliopoulos, Dgm: A deep learning algorithm for solving partial differential equations, *J. Comput. Phys.* **375**, 1339 (2018).
- [21] M. Magill, F. Qureshi, and H. W. de Haan, Neural networks trained to solve differential equations learn general representations, in *Advances in Neural Information Processing Systems (NeurIPS 2018)*, Vol. 31 (Curran Associates, Inc., 2018).
- [22] H. Li, Q. Zhai, and J. Z. Y. Chen, Neural-network-based multi-state solver for a static Schrödinger equation, *Phys. Rev. A* **103**, 032405 (2021).
- [23] K. Hornik, Approximation capabilities of multilayer feedforward networks, *Neural Networks* **4**, 251 (1991).
- [24] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Physics-informed machine learning, *Nat. Rev. Phys.* **3**, 422 (2021).
- [25] S. Wang and P. Perdikaris, Deep learning of free boundary and Stefan problems, *J. Comput. Phys.* **428**, 109914 (2021).
- [26] E. Kharazmi, M. Cai, X. Zheng, Z. Zhang, G. Lin, and G. E. Karniadakis, Identifiability and predictability of integer- and fractional-order epidemiological models using physics-informed neural networks, *Nat. Comput. Sci.* **1**, 744 (2021).
- [27] M. Angeli, G. Neofotistos, M. Mattheakis, and E. Kaxiras, Modeling the effect of the vaccination campaign on the COVID-19 pandemic, *Chaos, Solitons Fractals* **154**, 111621 (2022).
- [28] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, Fourier neural operator for parametric partial differential equations, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2021), <https://openreview.net/forum?id=c8P9NQVtmnO>.
- [29] A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, in *Proceedings of the Conference NeurIPS* (Curran Associates, Inc., 2021).
- [30] M. J. Gander, 50 years of time parallel time integration, in *Multiple Shooting and Time Domain Decomposition Methods*, edited by T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, Contributions in Mathematical and Computational Sciences, Vol. 9 (Springer, Cham, 2015), pp. 69–114.
- [31] S. Gunther, L. Ruthotto, J. B. Schroder, E. C. Cyr, and N. R. Gauger, Layer-parallel training of deep residual neural networks, *SIAM J. Math. Data Sci.* **2**, 1 (2020).
- [32] S. Greydanus, M. Dzamba, and J. Yosinski, Hamiltonian neural networks, in *Advances in Neural Information Processing Systems*, Vol. 32 (Curran Associates, Inc., 2019), pp. 15379–15389.
- [33] T. Bertalan, F. Dietrich, I. Mezic, and I. G. Kevrekidis, On learning Hamiltonian systems from data, *Chaos* **29**, 121107 (2019).
- [34] A. Choudhary, J. F. Lindner, E. G. Holliday, S. T. Miller, S. Sinha, and William L. Ditto, Physics-enhanced neural networks learn order and chaos, *Phys. Rev. E* **101**, 062207 (2020).
- [35] S. A. Desai, M. Mattheakis, and S. J. Roberts, Variational integrator graph networks for learning energy-conserving dynamical systems, *Phys. Rev. E* **104**, 035310 (2021).
- [36] C.-D. Han, B. Glaz, M. Haile, and Y.-C. Lai, Adaptable Hamiltonian neural networks, *Phys. Rev. Res.* **3**, 023156 (2021).
- [37] H. Zhang, H. Fan, L. Wang, and X. Wang, Learning Hamiltonian dynamics with reservoir computing, *Phys. Rev. E* **104**, 024205 (2021).
- [38] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia, Hamiltonian graph networks with ODE integrators, [arXiv:1909.12790](https://arxiv.org/abs/1909.12790).
- [39] P. Toth, D. J. Rezendes, A. Jaegle, S. Racanière, A. Botev, and I. Higgins, Hamiltonian generative networks, in *Proceedings of the International Conference on Learning Representations (ICLR)* (2020), <https://openreview.net/forum?id=HJenn6VFvB>.
- [40] S. A. Desai, M. Mattheakis, D. Sondak, P. Protopapas, and S. J. Roberts, Port-Hamiltonian neural networks for learning explicit time-dependent dynamical systems, *Phys. Rev. E* **104**, 034312 (2021).
- [41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, Automatic differentiation in PyTorch, in *Proceedings of the Conference NeurIPS* (Curran Associates, Inc., 2017).
- [42] <https://github.com/mariosmat/hamiltonianNNetODEs>.
- [43] E. Noether, Invariant variation problems, *Trans. Theor. Stat. Phys.* **1**, 186 (1971).
- [44] B. J. Leimkuhler and Robert D Skeel, Symplectic numerical integrators in constrained Hamiltonian systems, *J. Comput. Phys.* **112**, 117 (1994).
- [45] L. Ziyin, T. Hartwig, and M. Ueda, Neural networks fail to learn periodic functions and how to fix it, in *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33 (Curran Associates, Inc., 2020), pp. 1583–1594.
- [46] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference for Learning Representations, San Diego* (2015).
- [47] T. E. Oliphant, Python for scientific computing, *Computing in Science & Engineering*, **9**, 10 (2007).
- [48] M. Hénon and C. Heiles, The applicability of the third integral of motion: Some numerical experiments, *Astron. J.* **69**, 73 (1964).
- [49] I. I. Shevchenko and A. V. Mel’nikov, Lyapunov exponents in the Hénon-Heiles problem, *JETP Lett.* **77**, 642 (2003).