

Port-Hamiltonian Neural Networks for Learning Explicit Time-Dependent Dynamical Systems

Anonymous Authors¹

Abstract

Learning the dynamics of complex physical systems requires models with well-chosen inductive biases. Recently, it was shown that neural networks designed to learn a Hamiltonian and exploit Hamilton’s equations significantly outperform existing approaches in predicting trajectories of autonomous systems that depend implicitly on time. Despite this success, many real world physical systems are non-autonomous and depend explicitly on time. In this paper, we address this challenge by embedding a modified Port-Hamiltonian into our neural networks that extends the general Hamiltonian to capture damped and forced systems alike. We show that our network can learn the dynamics of complex physical systems such as a damped and driven oscillator, the Duffing equation in the chaotic regime, as well as a forced relativistic particle in a potential well. In all these settings, we show that our network can not only outperform existing baselines but can also accurately recover the underlying Hamiltonian, force and damping terms. We are also able to show that our network can be used to study the chaotic Duffing equation for long periodic trajectories and visually recover its Poincaré section - a crucial map in identifying chaotic trajectories.

1. Introduction

Neural networks (NNs), as universal function approximators (Hornik et al., 1989), have shown resounding success across a host of domains including image segmentation (He et al., 2018), machine translation (Devlin et al., 2019) and even material property predictions (Toussaint et al., 2018; Yao et al., 2018). However, their performance in learning dynam-

ical systems that obey physical laws has often been limited (Greydanus et al., 2019; Pukrittayakamee et al., 2009). New research in *scientific machine learning* - a field that tackles scientific problems with domain-specific machine learning methods, is paving a way to address this challenge. More specifically, it has been shown that physically-informed inductive biases embedded in networks, such as Hamiltonian mechanics (Mattheakis et al., 2020; Greydanus et al., 2019), Lagrangians (Cranmer et al., 2020; Lutter et al., 2019), Ordinary Differential Equations (ODEs) (Chen et al., 2018), physics-informed networks (Raissi et al., 2017) and Graph Neural Networks (Battaglia et al., 2016; Sanchez-Gonzalez et al., 2019) can significantly improve learning over vanilla neural networks in complex physical domains. Furthermore, it has also been shown that such systems can be adapted to learn trajectories of controlled systems (Lutter et al., 2019; Zhong et al., 2020).

Despite this extensive research, there is no method that accounts for explicit-time dependence, a crucial component in forced dynamical systems. Additionally, many of the existing approaches do not outline how to account for energy dissipation. In the present work, we outline how a Port-Hamiltonian formalism can be used to represent a forced and damped dynamical system. We then show how the underlying structure of this formalism can be embedded into a neural network in order to learn the dynamics of explicit time-dependent physical systems for which the forcing and damping terms are unknown apriori. We extensively benchmark our network on a range of tasks including the simple mass-spring system undergoing damping and complex forcing, the Duffing equation in both the non-chaotic and chaotic regimes as well as the Duffing equation for a relativistic particle. Our proposed network consistently outperforms other approaches while recovering both the driving force and the damping coefficient - an important result in system identification. Furthermore, we show that our network can be used to visually recover the Poincaré Section of the Duffing equation in a chaotic regime, a deeply promising result that can help us identify and understand chaotic trajectories (see Fig. 1).

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

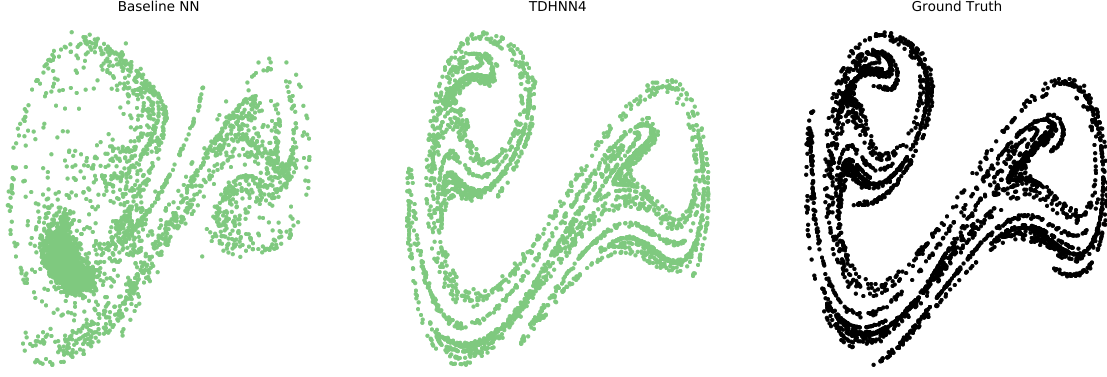


Figure 1. Poincaré Sections of a Duffing oscillator in a chaotic regime. Both Baseline NN and pHNN are trained for 20000 iterations with 2000 data points. pHNN significantly outperforms Baseline NN at recovering the ground truth Poincaré section of a test point not in the training set.

2. Background

2.1. Hamiltonian Neural Networks

Recently, the authors of (Greydanus et al., 2019) demonstrate that the dynamics of an energy conserving autonomous system can be accurately learned by guiding a neural network to predict a Hamiltonian - an important representation of a dynamical system that generalizes classical mechanics. Considering dynamical systems of M objects, the Hamiltonian \mathcal{H} is a scalar function of a position vector $\mathbf{q}(t) = (q_1(t), q_2(t), \dots, q_M(t))$ and momentum vector $\mathbf{p}(t) = (p_1(t), p_2(t), \dots, p_M(t))$ that obeys Hamilton's equations,

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

where $\dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt}$ and $\dot{\mathbf{p}} = \frac{d\mathbf{p}}{dt}$. In the following, we drop the time dependence where the context makes it clear.

Using this formalism, (Greydanus et al., 2019) show that a neural network, parametrized by θ , can be used to learn a scalar output Hamiltonian $H_\theta(\mathbf{q}, \mathbf{p})$ given \mathbf{q} and \mathbf{p} as inputs to the network. Using this predicted Hamiltonian, it is possible to recover the time derivatives of the input by simply differentiating (via autograd) the Hamiltonian with respect to its inputs. The derivatives of the Hamiltonian $\left[\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}\right]$ provide us with time derivatives of the input state vector $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$. Therefore, once trained, this network yields an approximate time derivative generator for the state $[\mathbf{q}, \mathbf{p}]$ and as such can be used in a numerical integrator to evolve an initial condition $[\mathbf{q}_t, \mathbf{p}_t]$, as in Eq.2.

$$(\mathbf{q}, \mathbf{p})_{t+1} = (\mathbf{q}, \mathbf{p})_t + \int_t^{t+1} \left[\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \right] dt. \quad (2)$$

In addition, the vector field $\mathbf{S} = \left[\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \right]$ is a symplectic gradient. This means that \mathcal{H} remains constant as long as the

state vectors are integrated along \mathbf{S} . This result links the Hamiltonian with the total energy of the system such that $\mathcal{H}(\mathbf{q}, \mathbf{p}) = E_{\text{tot}}$ for many autonomous physical systems. Therefore, the Hamiltonian is a powerful inductive bias that can be utilized to evolve a physical state while maintaining energy conservation. It is precisely for this reason that HNN outperforms traditional approaches that directly predict the state time derivatives from the input state. However, this formulation does not readily generalize to damped or forced time-varying systems.

2.2. Port-Hamiltonians

The Port-Hamiltonian (van der Schaft, 2007; Acosta et al., 2005; Zheng et al., 2018; Cherifi, 2020) is a well studied formalism in control literature that allows us to generalize Hamilton's equations to incorporate energy dissipation and an external control input to a dynamical system. A recent method that embeds the Port-Hamiltonian in a neural network (Zhong et al., 2020) shows how to represent a general form for such a system. In (Zhong et al., 2020) the Port-Hamiltonian is represented as:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{p}} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} + \mathbf{D}(\mathbf{q}) \right) \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial \mathbf{q}} \\ \frac{\partial \mathcal{H}}{\partial \mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{G}(\mathbf{q}) \end{bmatrix} \mathbf{u}, \quad (3)$$

where $\mathbf{D}(\mathbf{q})$ is a matrix to account for damping and $\mathbf{G}(\mathbf{q})\mathbf{u}$ is an external control term where \mathbf{u} is a temporal control input and $\mathbf{G}(\mathbf{q})$ is a spatial matrix. \mathbf{I} is the identity matrix and $\mathbf{0}$ the zero matrix. In this setting, the vector $[\mathbf{q}, \mathbf{p}, \mathbf{u}]$ is provided as input to the system. The matrix $\mathbf{D}(\mathbf{q})$ is assumed to be semi-positive definite and $\mathbf{G}(\mathbf{q})$ is typically a non-linear scaling of \mathbf{q} . This general formalism allows us to collapse to a classical Hamiltonian system by simply setting \mathbf{D} and \mathbf{u} to zero. Typically, the Port Hamiltonian is used in control applications where explicit knowledge of the control term \mathbf{u} is known, however, in this paper we explore

an adaptation of this formalism in which we do not know the control term and make this a learnable function. We do this because in some applications, it might be useful to uncover/identify to underlying forcing term from the state vector where no explicit knowledge of the control input is available.

3. Related Work

While Hamiltonian mechanics presents one way to address learning dynamical systems, numerous recent methods highlight how incorporating physically-informed inductive biases into neural networks can improve learning.

Functional Priors

Functional priors, for example, embed the full functional form of an equation into the neural network. Physics-informed Neural Networks (PINNs) (Raissi et al., 2017; 2019) and Hamiltonian Nets (Mattheakis et al., 2020) are two such approaches that look at directly embedding the equations of motion into the loss function. While PINNs are data-driven approaches that rely on autograd to compute partial-derivatives of a hidden state, Hamiltonian Nets are data-independent approaches that pre-specify the full functional form of a system-specific Hamiltonian in the loss function.

Integrator Priors

Many recent methods have sought to use a NN to parametrize the time derivatives of a state vector and then use this learned parametrization in a numerical integrator such as a Runge-Kutta method. The authors of NeuralODE (Chen et al., 2018) show that embedding the integrator into the learning process induces a continuous depth neural network for large time-step predictions. As such, the number of gradient computations scales exponentially with the number of integrative time steps during backpropagation through the integrator. The authors of NeuralODE thus show that the adjoint method can be used to ensure linear scaling of the gradient computations. More recent work has looked at generalizing this approach to different topologies that classical NeuralODEs cannot represent (Dupont et al., 2019). Other work such as (Zhu et al., 2020), theoretically shows the importance of using symplectic integrators over Runge-Kutta methods to evolve Hamiltonian systems in NNs.

Graph based methods

In (Battaglia et al., 2016), the authors detail how a Graph Neural Network, designed to capture the relational structure of systems, can be used to learn interacting particle systems. This work has been exploited in numerous advances (Sanchez-Gonzalez et al., 2018; 2020; Cranmer et al., 2020)

and emphasizes how a relation inductive bias can significantly improve learning.

Lagrangian/Hamiltonian Methods

While (Cranmer et al., 2020) and (Greydanus et al., 2019) propose to explicitly learn Hamiltonian and Lagrangian functions, (Lutter et al., 2019) shows how one can exploit the Lagrangian equation to learn both a function which can predict the generalized forces as well as use generalized forces and coordinate information to predict the next state. The authors show that their inverse model can accurately predict a controlled double pendulum, which is pertinent for controlled robots.

Latest Advances

Recent work studies Constrained HNN (Finzi et al., 2020), an approach that looks at enforcing Cartesian coordinates in Hamiltonians and Lagrangians with holonomic constraints. The paper emphasises that such a framework works significantly better at learning deeply complex physical systems such as the 5-pendulum.

Despite these significant breakthroughs, there is no existing method that investigates explicit time-dependence and damping in such systems, elements that are often found in real world problems. As such, we outline a novel technique to address this challenge.

4. Method

4.1. Theory

Our method, Time Dependent Port Hamiltonian NN (pHNN), takes the form of the Port-Hamiltonian with two modifications. First, our approach exploits the fact that many damped systems consist of a non-zero, state-independent damping term in the lower right quadrant, so we replace the damping matrix $\mathbf{D}(\mathbf{q})$ with a state independent lower right matrix \mathbf{N} . Secondly, in order to generalize to time dependent forcing, we replace $\mathbf{G}(\mathbf{q})\mathbf{u}$ with $\mathbf{F}(t)$. The resulting representation, as highlighted in Eq.(4), is now general enough to handle many well-known forced systems but also specific enough to tackle learning in physical domains of practical importance such as the Duffing equation.

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{p}} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{N} \end{bmatrix} \right) \begin{bmatrix} \frac{\partial \mathcal{H}}{\partial \mathbf{q}} \\ \frac{\partial \mathcal{H}}{\partial \mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{F}(t) \end{bmatrix}. \quad (4)$$

The architecture for our method as well as existing approaches are shown in Fig.2. Classic NN Fig.2(a) and Hamiltonian NN Fig.2(c) (Greydanus et al., 2019) simply take position and momentum as inputs and are designed to yield the time derivatives of the input, with HNN learning an

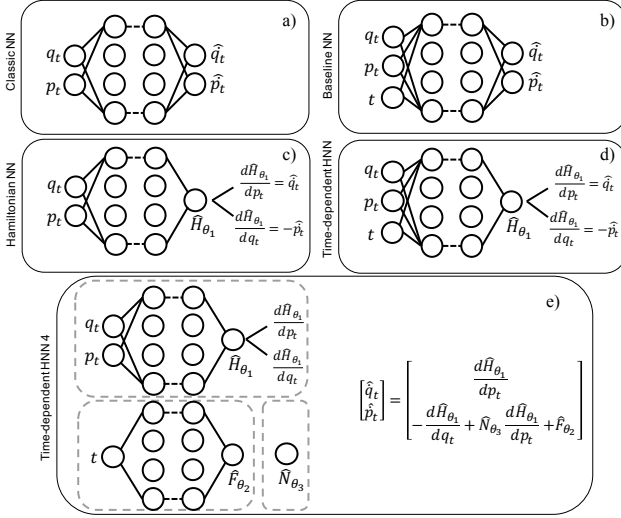


Figure 2. Architectures used to learn dynamics in this paper. The naive extension of classic NN (a) and Hamiltonian NN (c) is to incorporate time as an additional input variable (b and d). Our innovation, which exploits Port-Hamiltonians, explicitly learns the force F_{θ_2} as well as the damping term N_{θ_3} (e).

intermediary Hamiltonian and employing backpropagation to compute the output. One natural way to extend these architectures for time-varying non-autonomous systems is to simply include time as an additional input. This readily gives rise to Baseline NN (Fig.2(b)) and Time-Dependent HNN (Fig.2(d)). While we will see that both these approaches are reasonable, they do not help us identify the underlying force or damping terms, nor are they consistently performant across the applications we investigate. Our network Fig.2(e) goes beyond this naive approach and embeds the Port-Hamiltonian. The network is designed to learn the Hamiltonian, \mathcal{H} , the driving force, \mathbf{F} , and the damping term, \mathbf{N} , and combine these to obtain estimates of the state time derivatives.

4.2. Training

To train the network, we feed in $[\mathbf{q}, \mathbf{p}, t]$ to our model. The first component, \mathcal{H}_{θ_1} , consists of three hidden layers designed to predict \mathcal{H} from $[\mathbf{q}, \mathbf{p}]$ input data. The second component, \mathbf{N}_{θ_3} , consists of a single weight parameter (i.e. θ_3 is a single node) designed to learn the damping. The third neural-network consists of three hidden layers designed to predict $\mathbf{F}(t)$. The output of each component is combined as per Eq.4 to obtain predictions of the state time derivatives $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$. Using these predictions, we define our loss function as:

$$\mathcal{L} = \|\hat{\mathbf{q}}_t - \dot{\mathbf{q}}_t\|_2^2 + \|\hat{\mathbf{p}}_t - \dot{\mathbf{p}}_t\|_2^2 + \lambda_F \|\hat{\mathbf{F}}_{\theta_2}(t)\|_1 + \lambda_N \|\hat{\mathbf{N}}_{\theta_3}\|_1. \quad (5)$$

The loss function minimizes the loss between the predicted and ground truth state time derivatives with a squared er-

ror loss. In addition, the forcing and damping terms are added to the loss with an L1 penalty when using pHNN. The latter choice is made to encourage the networks to learn simpler models. We also find, during experimentation, that this choice is superior to L2 in enabling us to identify the ground truth force and damping terms. The choice of hyper-parameters λ_F and λ_N is made via a grid search across $[10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}]$, for each parameter, that generates the lowest validation loss (details in Appendix D). For our experiments we use 200 nodes per layer and find that most activations, including tanh, sin and cos yield comparable results.

4.2.1. TRAINING DATASET

Training data is generated using a 4th order Runge-Kutta integrator. Several simulations were run with different initial conditions for $t \in [0, T_{\max}]$. At each iteration, the next state as well as the state derivatives are recorded. This is typically done for multiple initial conditions and then passed to the network for training. Details for how initial conditions are sampled are described independently for each system in the results section.

4.3. Testing

Once trained, each of the networks in Fig.2 can approximate $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$. As such, they can be used in a scientific integrator such as a Runge-Kutta 4th order method to integrate initial conditions not in the training dataset from $t = 0$ to $t = T_{\max}$ which we refer to as a state rollout. We measure the performance of the network by comparing the predicted state rollout with the ground truth. We do this by computing a mean squared error across the state variables and across time. Therefore:

$$\text{MSE}_{\text{state}} = \frac{1}{2N} \left(\sum_{i=1}^N (\mathbf{q}_i - \hat{\mathbf{q}}_i)^2 + \sum_{i=1}^N (\mathbf{p}_i - \hat{\mathbf{p}}_i)^2 \right), \quad (6)$$

and

$$\text{MSE}_{\text{energy}} = \frac{1}{N} \sum_{i=1}^N (\mathcal{H}(\mathbf{q}_i, \mathbf{p}_i) - \mathcal{H}(\hat{\mathbf{q}}_i, \hat{\mathbf{p}}_i))^2 \quad (7)$$

where $N = T_{\max}/\Delta t$, with Δt as the integrative step. We typically compute these terms for multiple different initial conditions during inference and average across them (see Fig.3(c)).

4.4. Procedure

1. Obtain state variable data $[\mathbf{q}, \mathbf{p}, t]$ and time derivatives $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$ from trajectories of a given system.
2. Provide state-variable information to pHNN which learns a Hamiltonian, force and damping term to predict $[\hat{\mathbf{q}}, \hat{\mathbf{p}}]$.
3. Minimize predictions with ground truth time-derivatives as in Eqn.(5).

4. Once trained, use the network as a time-derivative generator in a scientific integrator to integrate a set of initial conditions $[\mathbf{q}_0, \mathbf{p}_0]$ to $[\mathbf{q}_T, \mathbf{p}_T]$.

5. Results

We benchmark the performance of our models on numerous datasets that cover time-independent systems to complex chaotic forced systems. The results are presented in order of increasing complexity from a model perspective.

5.1. Simple Mass-Spring System

We begin our analysis with a simple mass-spring system, obeying Hooke’s Law from classical physics with no force or damping. The Hamiltonian is,

$$\mathcal{H} = \frac{1}{2}kq^2 + \frac{1}{2m}p^2. \quad (8)$$

where k is the spring constant and m is the mass of the object. Note, the position and moment are scalars.

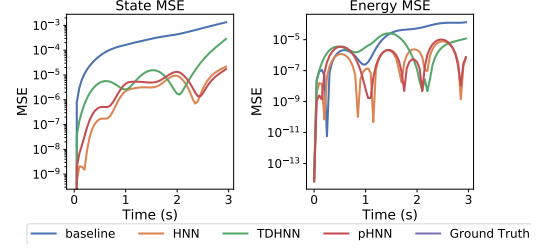
Training: Without loss of generalization, we set k and m to 1 for our experiments. We randomly sample 25 initial training conditions $[q_0, p_0]$ that satisfy $q_0^2 + p_0^2 = r_0^2$ where $1 \leq r_0 \leq 4.5$ which corresponds to sampling initial conditions with energies between 1 and 4.5. We integrate each using a 4th order RK method with $\Delta t = 0.05$ and $T_{\max} = 3.05$.

Test: At test time, we sample 25 initial conditions, not in the training set, and evolve them to $T_{\max} = 3.05$. We investigate this simple system to show that learning a separate, regularized forcing term results in much better state and energy predictions in comparison to TDHNN and the Baseline NN which may not readily ignore the time variable fed in. Learning a separate forcing term and regularizing it keeps the time component independent of the Hamiltonian and therefore allows us to closely match the performance of classical time-implicit HNN.

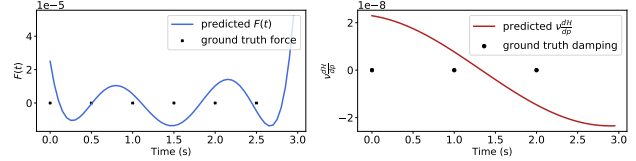
5.2. Damped Mass-Spring System

We extend the simple mass-spring system to include a damping term that reduces the initial energy of the system over time. The inclusion of this term violates energy conservation and therefore we cannot write a scalar Hamiltonian for such a system (see Appendix A.1 for details). Knowing this apriori already gives us an indication that HNN will perform poorly on such a system. The damped mass-spring system is typically used as a model to design shock absorbers for cars in order to ensure a smooth and safe ride.

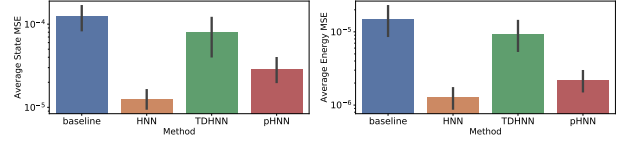
Training: We have 20 initial training conditions, with position and momentum uniformly sampled in $[-1, 1]^2$. Each trajectory is evolved until $T_{\max} = 30.1$ with a $\Delta t = 0.1$.



(a) State and energy MSE as a function of time of an initial condition in the test set.



(b) Learnt force and damping terms by pHNN



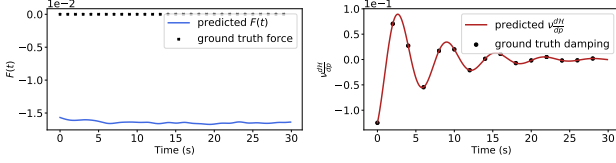
(c) State and energy MSE averaged across 25 initial test states

Figure 3. The simple mass-spring system has no explicit time dependence. We see that pHNN can almost recover the dynamics as well as in HNN. The baseline NN and TDHNN are unable to achieve the same test state error as they are only reliable for time steps that are within the training regime. While pHNN does learn a non-zero force and damping term, their contribution to $\frac{dp}{dt}$ is small.

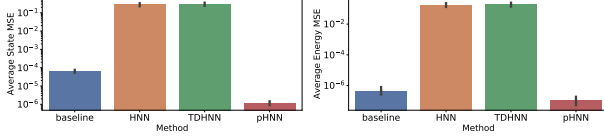
We fix the damping coefficient $\nu = 0.3$ without loss of generality to other coefficients.

Test: At inference, we compute the average rollout MSE of 25 unseen initial conditions sampled in the same manner as the training data, and report the average state and energy rollout MSE (see Fig.4).

Firstly, we can see that both baseline NN and pHNN recover the dynamics well, whereas HNN (as expected) and TDHNN struggle to learn the damping. The main reason for this latter result is that there is no direct way of writing a scalar Hamiltonian with damping so forcing a network to do so results in unreliable test results. This result is important because it informs us that a partial inductive bias is not good at generalizing. Secondly, we see that pHNN learns a non-zero, oscillating force. This can happen within our system because the network is not guaranteed to find the global optima and it is indeed possible for information from the Hamiltonian, \mathcal{H} , to leak into the force and vice-versa. This is because there is an identifiability challenge in attempting to uncover the damping and force exclusively from the state data we provide. In spite of this challenge, we find that our networks converge to reasonable forces and damping terms



(a) Learnt Force and Damping terms by pHNN



(b) State and energy MSE averaged across 25 initial test states

Figure 4. Damped mass-spring setting: The baseline NN and pHNN recover the underlying dynamics well. pHNN is also able to accurately learn the damping coefficient since the predicted damping is indistinguishable from the ground truth.

that can inform us of the underlying dynamics as well as help us make predictions. Thirdly, in this particular example, we do see that while the force and damping are non-zero, the force is 5 orders of magnitude smaller than $\frac{dp}{dt}$ and so its contribution is negligible.

5.3. Forced Mass-Spring System

Typically, while we cannot write the Hamiltonian for a damped system, we can write one for a forced system. Here, we study two types of forced mass-spring systems. The first has the following Hamiltonian form:

$$\mathcal{H} = \frac{1}{2}kq^2 + \frac{1}{2m}p^2 - qF_0 \sin(\omega t). \quad (9)$$

The second has the Hamiltonian:

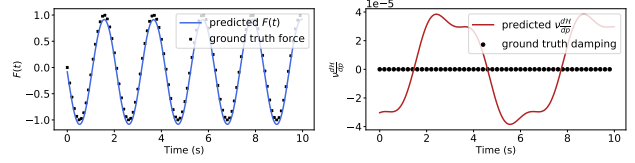
$$\mathcal{H} = \frac{1}{2}kq^2 + \frac{1}{2m}p^2 - qF_0 \sin(\omega t) \sin(2\omega t). \quad (10)$$

The forced mass-spring system is typically used to study resonance and plays an important role in numerous applications including music, bridge design and molecular excitation making it an important system to investigate.

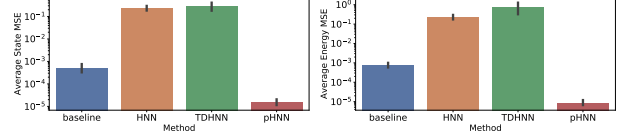
Training: In both settings, we use 20 initial conditions, where the initial state $[q_0, p_0]$ is sampled such that $q_0^2 + p_0^2 = r_0^2$ where $1 \leq r_0 \leq 4.5$. The states are rolled out to $T_{\max} = 10.01$ at a $\Delta t = 0.01$. k , m and F_0 are all set to 1 without loss of generality. The forcing frequency ω is set to 3.

Test: At inference, we compute the rollout of 25 unseen initial conditions, and report the average state and energy rollout MSE in the Fig. 5 and Fig. 6.

We study both systems here to illustrate that while the baseline NN is able to do relatively well in comparison to pHNN

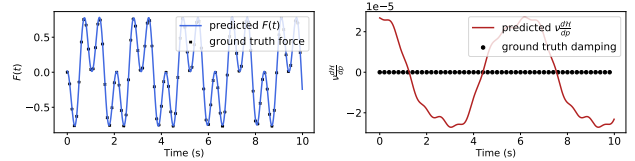


(a) Learnt Force and Damping terms by pHNN

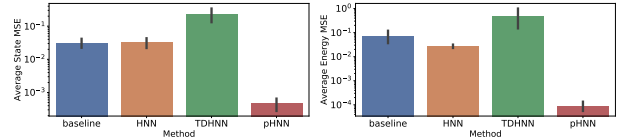


(b) Rollout state and energy MSE averaged across 25 initial test states

Figure 5. Forced mass-spring (I): HNN cannot learn the underlying dynamics as it has no explicit-time dependence. pHNN shows the best performance as it explicitly learns a time-dependent force.



(a) Learnt Force and Damping terms by pHNN



(b) Rollout state and energy MSE averaged across 25 initial test states

Figure 6. Forced mass-spring (II): The time dependent force here is non-trivial, but pHNN shows it can recover it.

with a simple force, a more complex force significantly hurts its performance in terms of state/energy MSE. More importantly, we see that for both systems pHNN can recover the ground truth force quite precisely and while they both learn a non-zero damping term, their contribution to $\frac{dp}{dt}$ is negligible.

5.4. Duffing Equation

We now investigate the Duffing equation that brings both forcing and damping to the mass-spring system. We do note that the unforced and undamped regular Hamiltonian \mathcal{H}_{reg} of the Duffing system is:

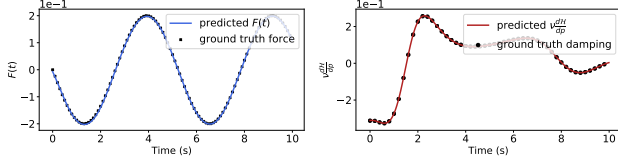
$$\mathcal{H}_{reg} = \frac{p^2}{2m} + \alpha \frac{q^2}{2} + \beta \frac{q^4}{4}. \quad (11)$$

Unlike the simple mass-spring system, the Duffing equation has an additional quartic function of q . This means

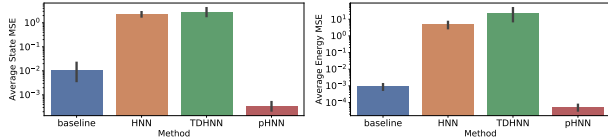
that the shape of the potential function can be a double-well or a single well based on the coefficients α and β . The general Duffing equation combines both the force and damping terms discussed previously with $\frac{d\mathcal{H}_{reg}}{dq}$. Typically the equation is written as:

$$\ddot{q} = -\delta\dot{q} - \alpha q - \beta q^3 + \gamma \sin(\omega t). \quad (12)$$

A combination of parameters $\alpha, \beta, \delta, \gamma, \omega$ will make the Duffing system either chaotic or non-chaotic. We study both regimes.

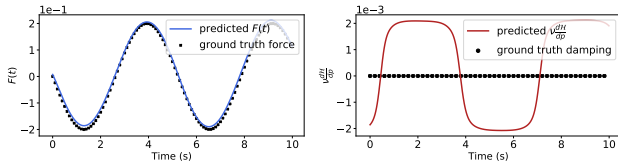


(a) Learnt Force and Damping terms of pHNN

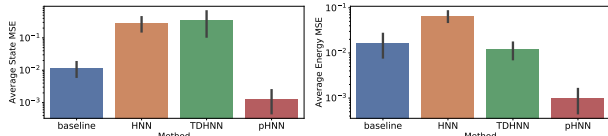


(b) State and energy MSE averaged across 25 initial test states

Figure 7. Duffing equation (non-chaotic): pHNN significantly outperforms the other methods and is able to extract the ground truth force and damping coefficient.



(a) Learnt Force and Damping terms of pHNN



(b) Rollout state and energy MSE averaged across 25 initial test states

Figure 8. Learned dynamics of a relativistic Duffing system.

5.4.1. NON-CHAOTIC REGIME

Given a set of initial parameters for the Duffing equation: $\alpha = -1, \beta = 1, \delta = 0.3, \gamma = 0.2, \omega = 1.2$ we can obtain training data for the non-chaotic regime of the Duffing equation.

Training: We uniformly sample initial conditions in $[-1, 1]^2$. We use 25 initial conditions for training, rolled

out to $T_{\max} = 10.01$ and $\Delta t = 0.01$ for the non-chaotic regime.

Test: We integrate 25 unseen initial conditions at inference using the same conditions as above and evaluate both energy and state MSE (see Fig.7).

In addition to learning the force and damping terms accurately, inspecting the predicted Hamiltonian over a 2-D grid of position and momentum values shows that pHNN can also accurately learn the functional form of \mathcal{H}_{reg} (see Fig.9).

5.4.2. CHAOTIC REGIME

In the chaotic regime we use the following parameters: $\alpha = 1, \beta = 1, \delta = 0.1, \gamma = 0.39, \omega = 1.4$.

Training: 20 initial conditions, sampled uniformly in $[-1, 1]^2$ each rolled out for one period $T = 2\pi/\omega$ where $\Delta t = T/100$. This results in 2000 training points.

Test: We test our system by assessing whether it is visually able to recover the ground truth Poincaré section of an initial condition. The Poincaré map of a trajectory is measured by plotting the position and momentum values at regular intervals governed by the period of the forcing term. For example, a simple mass-spring system will generate a single point in phase space when measured at regular intervals. To visually assess the performance of our network through a Poincaré map, we test the system on a single initial condition not in the training set rolled out to $T_{\max} = 18,000$ with the same Δt as training. In order to integrate our system to such a large T_{\max} for this example we work under the assumption that we have explicit knowledge of the period, and as such, we modulo the time variable with the period. This is necessary as the models are not explicitly trained on time steps beyond $2\pi/\omega$. Our results are visually presented in Fig.1. The outcome suggests that pHNN can indeed be used to identify chaos even after being trained with only a few data points from a chaotic trajectory. We believe this is a deeply powerful result in helping us identify chaotic trajectories.

5.5. Relativistic System

We also go beyond the classical approach and investigate the Duffing equation in a relativistic framework. The Hamiltonian under consideration is:

$$\mathcal{H} = c\sqrt{p^2 + m_0^2 c^2} + \frac{\alpha}{2}q^2 + \frac{\beta}{4}q^4 - q\gamma \sin(\omega t), \quad (13)$$

where c , the speed of light is typically set to 1. For simplicity, we also set the rest mass $m_0 = 1$ though our framework naturally accounts for other values.

Training: We train on 25 initial conditions, sampled in $[0, 2]^2$ uniformly. $T_{\max} = 20.01$ and $\Delta t = 0.01$. We set

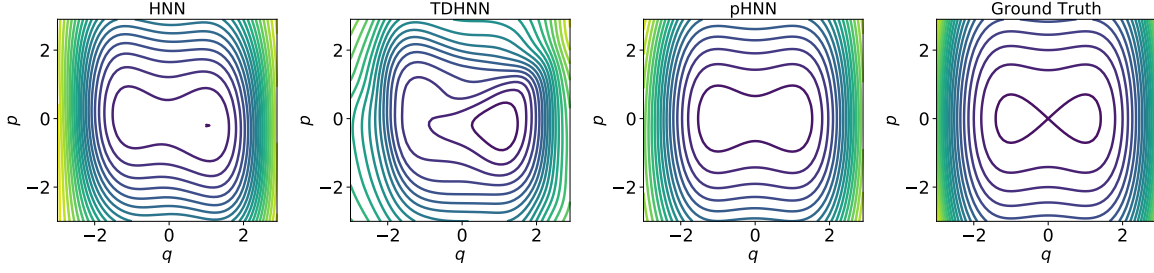


Figure 9. Learnt \mathcal{H}_{reg} components across methods in the non-chaotic Duffing setting. HNN and TDHNN learn distorted Hamiltonians that strongly depend on the input time-variable.

Table 1. State and Energy MSE for each method averaged across 25 initial test points

	Baseline		HNN		TDHNN		pHNN	
	State	Energy	State	Energy	State	Energy	State	Energy
Mass Spring	$1.22\text{E-}4 \pm 1.06\text{E-}4$	$1.56\text{E-}5 \pm 1.84\text{E-}5$	$1.33\text{E-}5 \pm 8.05\text{E-}6$	$1.30\text{E-}6 \pm 1.31\text{E-}6$	$7.84\text{E-}5 \pm 1.02\text{E-}4$	$9.26\text{E-}6 \pm 1.13\text{E-}5$	$2.91\text{E-}5 \pm 2.57\text{E-}5$	$2.33\text{E-}6 \pm 2.41\text{E-}6$
Damped Mass-Spring	$6.4\text{E-}5 \pm 2.904\text{E-}5$	$4.68\text{E-}7 \pm 7.51\text{E-}7$	$2.81\text{E-}1 \pm 1.37\text{E-}1$	$1.81\text{E-}1 \pm 1.37\text{E-}1$	$3.01\text{E-}1 \pm 1.43\text{E-}1$	$1.92\text{E-}1 \pm 1.42\text{E-}1$	$1.74\text{E-}6 \pm 7.26\text{E-}7$	$1.08\text{E-}7 \pm 1.73\text{E-}7$
Forced (I)	$5.21\text{E-}4 \pm 5.74\text{E-}4$	$7.77\text{E-}4 \pm 5.52\text{E-}4$	$2.36\text{E-}1 \pm 1.43\text{E-}1$	$2.34\text{E-}1 \pm 1.70\text{E-}1$	$2.91\text{E-}1 \pm 3.06\text{E-}1$	$7.61\text{E-}1 \pm 1.23$	$1.53\text{E-}5 \pm 1.16\text{E-}5$	$9.21\text{E-}6 \pm 7.11\text{E-}6$
Forced (II)	$3.14\text{E-}2 \pm 2.54\text{E-}2$	$7.00\text{E-}2 \pm 1.14\text{E-}1$	$3.11\text{E-}2 \pm 2.67\text{E-}2$	$2.70\text{E-}2 \pm 1.12\text{E-}2$	$2.22\text{E-}1 \pm 2.82\text{E-}1$	$4.68\text{E-}1 \pm 1.06$	$4.51\text{E-}4 \pm 4.85\text{E-}4$	$8.91\text{E-}5 \pm 1.01\text{E-}4$
Duffing	$1.11\text{E-}2 \pm 2.43\text{E-}2$	$8.93\text{E-}4 \pm 8.75\text{E-}4$	2.31 ± 1.21	4.90 ± 5.30	2.91 ± 3.01	$2.43\text{E}1 \pm 5.11\text{E}1$	$3.50\text{E-}4 \pm 3.54\text{E-}4$	$5.32\text{E-}5 \pm 5.38\text{E-}5$
Relativistic Duffing	$5.85\text{E-}2 \pm 7.05\text{E-}2$	$3.61\text{E-}2 \pm 4.76\text{E-}2$	1.39 ± 1.66	$2.37\text{E-}1 \pm 1.55\text{E-}1$	$5.9\text{E-}1 \pm 1.44$	$4.39\text{E-}2 \pm 5.34\text{E-}2$	$4.96\text{E-}3 \pm 8.61\text{E-}3$	$1.28\text{E-}3 \pm 1.45\text{E-}3$

parameters such that $\alpha = 1, \beta = 1, \delta = 0, \gamma = 0.2, \omega = 1.2$.

Test: Using the same parameters as training, we rollout 25 unseen initial conditions and compute their state/energy MSE.

6. Discussion

We have shown that pHNN can not only outperform other approaches in learning complex physical systems (see Table.1) but can also help us recover the underlying force and damping terms of non-autonomous systems. One challenge in achieving this result is fine-tuning the λ_F and λ_N regularisation coefficients for the force and damping. In the Duffing setting where we have both terms, it is possible to learn a shifted force i.e. $F = F_0 + \epsilon$. The reason this is possible is because the state-vectors \mathbf{q}, \mathbf{p} do not provide enough information to simultaneously identify both the Hamiltonian and the force. As such it is possible that our hyper-parameter selection routine will pick coefficients that maintain a force or damping term in settings where they might not exist. However, we do find that in general a reasonable force and damping term are learnt so as to reveal the underlying dynamics.

In many settings we find that the classic Hamiltonian Neural Network approach or its time-dependent variant simply do not perform as well as pHNN or baseline. We believe this is predominantly due to the fact that HNN and TDHNN attempt to enforce Hamilton’s equations in settings where they cannot be satisfied by simply learning a single scalar Hamiltonian. In fact, this is perhaps the main reason why baseline does well because it does not enforce any structure.

While it may be argued that the method is constrained to learning and prediction within the training time horizon, we believe our method is still versatile at informing us of periodic forcing (since we can inspect the force over time). This in turn can readily be used to renormalize the time variable at periodic intervals to integrate the system beyond the training time as we show in Fig.1.

7. Conclusion

We have shown that learning the dynamics of time-dependent non-autonomous systems can be achieved with pHNN, a versatile neural network embedded with a Port-Hamiltonian designed to recover the underlying Hamiltonian, damping and forcing terms. We show that such an approach, unique in its own right, outperforms naive extensions of existing methods. We believe this work forms a strong basis for further advances in learning complex systems including, but not limited to, chemical bond forces and controlled dynamics without explicit knowledge of the force or damping. Future work in this direction is likely to explore how such a port Hamiltonian approach can be used for many-body systems with more complex forcing.

References

- Acosta, J. A., Ortega, R., Astolfi, A., and Mahindrakar, A. D. Interconnection and damping assignment passivity-based control of mechanical systems with underactuation degree one. *IEEE Transactions on Automatic Control*, 50 (12):1936–1955, December 2005. ISSN 1558-2523. doi: 10.1109/TAC.2005.860292. Conference Name: IEEE Transactions on Automatic Control.
- Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D., and Kavukcuoglu, K. Interaction Networks for Learning about Objects, Relations and Physics. *arXiv:1612.00222 [cs]*, December 2016. URL <http://arxiv.org/abs/1612.00222>. arXiv: 1612.00222.
- Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural Ordinary Differential Equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6571–6583. Curran Associates, Inc., 2018. URL <http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf>.
- Cherifi, K. An overview on recent machine learning techniques for Port Hamiltonian systems. *Physica D: Nonlinear Phenomena*, 411:132620, October 2020. ISSN 0167-2789. doi: 10.1016/j.physd.2020.132620. URL <http://www.sciencedirect.com/science/article/pii/S0167278919307705>.
- Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian Neural Networks. *arXiv:2003.04630 [physics, stat]*, March 2020. URL <http://arxiv.org/abs/2003.04630>. arXiv: 2003.04630.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. URL <http://arxiv.org/abs/1810.04805>. arXiv: 1810.04805.
- Dupont, E., Doucet, A., and Teh, Y. W. Augmented Neural ODEs. *arXiv:1904.01681 [cs, stat]*, October 2019. URL <http://arxiv.org/abs/1904.01681>. arXiv: 1904.01681.
- Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data. *arXiv:2002.12880 [cs, stat]*, May 2020. URL <http://arxiv.org/abs/2002.12880>. arXiv: 2002.12880.
- Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian Neural Networks. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 15379–15389. Curran Associates, Inc., 2019. URL <http://papers.nips.cc/paper/9672-hamiltonian-neural-networks.pdf>.
- He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. *arXiv:1703.06870 [cs]*, January 2018. URL <http://arxiv.org/abs/1703.06870>. arXiv: 1703.06870.
- Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL <http://www.sciencedirect.com/science/article/pii/0893608089900208>.
- Lutter, M., Ritter, C., and Peters, J. Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning. *arXiv:1907.04490 [cs, eess, stat]*, July 2019. URL <http://arxiv.org/abs/1907.04490>. arXiv: 1907.04490.
- Mattheakis, M., Sondak, D., Dogra, A. S., and Protopapas, P. Hamiltonian Neural Networks for solving differential equations. *arXiv:2001.11107 [physics]*, February 2020. URL <http://arxiv.org/abs/2001.11107>. arXiv: 2001.11107.
- Pukrittayakamee, A., Malshe, M., Hagan, M., Raff, L. M., Narulkar, R., Bukkapatnam, S., and Komanduri, R. Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks. *The Journal of Chemical Physics*, 130(13): 134101, April 2009. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.3095491. URL <http://aip.scitation.org/doi/10.1063/1.3095491>.
- Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv:1711.10561 [cs, math, stat]*, November 2017. URL <http://arxiv.org/abs/1711.10561>. arXiv: 1711.10561.
- Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL <http://www.sciencedirect.com/science/article/pii/S0021999118307125>.

- Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. *arXiv:1806.01242 [cs, stat]*, June 2018. URL <http://arxiv.org/abs/1806.01242>. arXiv: 1806.01242.
- Sanchez-Gonzalez, A., Bapst, V., Cranmer, K., and Battaglia, P. Hamiltonian Graph Networks with ODE Integrators. *arXiv:1909.12790 [physics]*, September 2019. URL <http://arxiv.org/abs/1909.12790>. arXiv: 1909.12790.
- Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W. Learning to Simulate Complex Physics with Graph Networks. *arXiv:2002.09405 [physics, stat]*, February 2020. URL <http://arxiv.org/abs/2002.09405>. arXiv: 2002.09405.
- Toussaint, M., Allen, K., Smith, K., and Tenenbaum, J. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018. ISBN 978-0-9923747-4-7. doi: 10.15607/RSS.2018.XIV.044. URL <http://www.roboticsproceedings.org/rss14/p44.pdf>.
- van der Schaft, A. Port-Hamiltonian systems: an introductory survey. In Sanz-Solé, M., Soria, J., Varona, J. L., and Verdera, J. (eds.), *Proceedings of the International Congress of Mathematicians Madrid, August 22–30, 2006*, pp. 1339–1365. European Mathematical Society Publishing House, Zuerich, Switzerland, May 2007. ISBN 978-3-03719-022-7. doi: 10.4171/022-3/65. URL <http://www.ems-ph.org/doi/10.4171/022-3/65>.
- Yao, K., Herr, J. E., Toth, D., Mckintyre, R., and Parkhill, J. The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics. *Chemical Science*, 9(8):2261–2269, 2018. ISSN 2041-6520, 2041-6539. doi: 10.1039/C7SC04934J. URL <http://xlink.rsc.org/?DOI=C7SC04934J>.
- Zheng, M., Yuan, T., and Huang, T. Time-Varying Impedance Control of Port Hamiltonian System with a New Energy-Storing Tank, November 2018. URL <https://www.hindawi.com/journals/complexity/2018/8134230/>. ISSN: 1076-2787 Pages: e8134230 Publisher: Hindawi Volume: 2018.
- Zhong, Y. D., Dey, B., and Chakraborty, A. Dissipative SymODEN: Encoding Hamiltonian Dynamics with Dissipation and Control into Deep Learning. *arXiv:2002.08860 [cs, eess, stat]*, April 2020. URL <http://arxiv.org/abs/2002.08860>. arXiv: 2002.08860.
- Zhu, A., Jin, P., and Tang, Y. Deep Hamiltonian networks based on symplectic integrators. *arXiv:2004.13830 [cs, math]*, April 2020. URL <http://arxiv.org/abs/2004.13830>. arXiv: 2004.13830.