TDHNN: Time-Dependent Hamiltonian Neural Networks

Shaan A. Desai[1,2] **Marios Mattheakis**[2] **David Sondak**[2] **Stephen Roberts**[1]
Machine Learning Research Group
University of Oxford [1]
School of Engineering Science
Harvard University [2]

Abstract

Deep neural-networks embedded with physically-informed priors demonstrate remarkable performance over vanilla neural-networks in accurately learning and predicting non-linear dynamical systems. Recently, it was shown that Hamiltonian Neural Networks (HNNs) - neural networks designed to conserve energy and exploit the Hamiltonian formalism, show strong and consistent performance in learning autonomous systems that depend implicitly on time. Here, we extend this framework to include an explicit time-dependence that generalizes the learning to non-autonomous dynamical systems. We achieve this generalisation 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 systems as complex as the duffing equation in the chaotic regime while maintaining comparable performance to HNNs in settings with no explicit time-dependence.

1 Introduction

Neural networks (NNs), as universal function approximators [8], have shown resounding success across a host of domains [7, 4, 16, 17]. However, their performance in learning physical systems has often been limited [6, 10]. New research in *scientific machine learning* - a branch that tackles scientific problems with domain-specific machine learning methods, is paving a way to address these challenges. For example, it has been shown that prior theoretical information embedded in networks, such as Hamiltonian mechanics [6], Lagrangians [3, 9], Ordinary Differential Equations (ODEs) [2], physics-informed networks [11] and Graph Neural Networks [1, 13] can significantly improve learning in complex physical domains. Furthermore, it has been shown that such systems can be adapted to learn trajectories of controlled dynamics [9, 18]. Despite this extensive research, there is still no method that accounts for explicit-time dependence, a crucial component in forced dynamical systems. Additionally, many of the existing approaches present no direct way to account for dissipation. Inspired by [18], we show that a Port-Hamiltonian - a formalism designed to account for forcing and dissipation, can be used to learn the dynamics of explicit time-dependent physical systems. We benchmark our network on a range of tasks from the simple mass-spring system to a chaotic duffing equation with a relativistic kinetic energy. Our network consistently outperforms other approaches while actively learning both the driving force and damping coefficient.

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

2 Background

2.1 Hamiltonian Neural Networks

Recently, [6] demonstrated that dynamic predictions through time can be improved using Hamiltonian Neural Networks (HNNs) which endow models with a Hamiltonian constraint. The Hamiltonian is an important representation of a dynamical system because it is one of two well-known approaches that generalizes classical mechanics. The Hamiltonian $\mathcal{H}$ is a scalar function of position $\mathbf{q} = (q_1, q_2, ..., q_M)$ and momentum $\mathbf{p} = (p_1, p_2, ..., p_M)$. The position and momentum are related to the Hamiltonian via Hamilton's equations (see Eqn.1.)

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \tag{1}$$

As a consequence, it is noted in [6] that by parametrizing the Hamiltonian with a neural network e.g. $H_\theta(\mathbf{q}, \mathbf{p})$ where θ represents the parameters of a deep

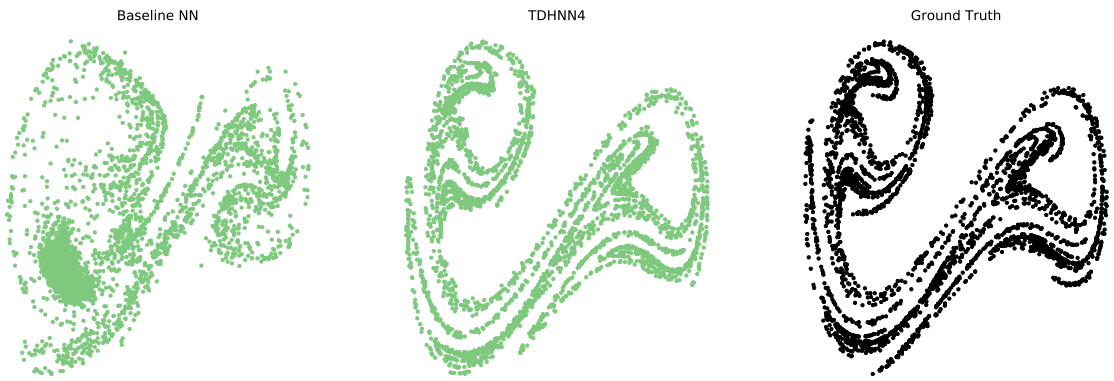


Figure 1: Poincaré Sections of a duffing oscillator in a chaotic regime. Both Baseline NN and TDHNN4 are trained for 20000 iterations with 2000 data points. TDHNN4 significantly outperforms Baseline NN at recovering the ground truth Poincaré section of a test point not in the training set.

neural network, one can easily obtain the system's dynamics by differentiating (via autograd) the Hamiltonian with its inputs. The derivatives of the Hamiltonian with respect to the input $\left[\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}\right]$ provide us with time derivatives of the state vector $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$ and can therefore be used to integrate an initial condition:

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

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 $\mathcal{H}(\mathbf{q}, \mathbf{p}) = E_{tot}$ for many physical systems. Therefore, the Hamiltonian is a powerful inductive bias that can be utilised to evolve a physical state while maintaining energy conservation.

Although this formalism is compact and powerful, it does not readily generalize to damped or forced system.

2.2 Port-Hamiltonians

The Port-Hamiltonian is a formalism that allow us to generalize Hamilton's equations to incorporate damping and forcing terms. The first method to embed the Port-Hamiltonian in a neural network [18] shows how to represent a general form for such a system. In [18] the Port-Hamiltonian is represented as:

$$\begin{bmatrix}\dot{\mathbf{q}} \\ \dot{\mathbf{p}}\end{bmatrix} = \left(\begin{bmatrix} 0 & \mathbf{I} \\ -\mathbf{I} & 0 \end{bmatrix} + \mathbf{D}_{\theta_2}(\mathbf{q}) \right) \begin{bmatrix} \frac{\partial \mathcal{H}_{\theta_1}}{\partial \mathbf{q}} \\ \frac{\partial \mathcal{H}_{\theta_1}}{\partial \mathbf{p}} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{g}(\mathbf{q}) \end{bmatrix} u \quad (3)$$

where $\mathbf{D}_{\theta_2}(\mathbf{q})$ is a matrix to account for damping and $g(q)u$ is the external constant control term.

This general formalism allows us to collapse to classical Hamiltonians by simply setting $\mathbf{D}$ and u to zero. Note here that the authors of [18] assume the vector $[\mathbf{q}, \mathbf{p}, u]$ is provided as input to the system. The matrix $\mathbf{D}_{\theta_2}(\mathbf{q})$ is assumed to be semi-positive definite and $g(\mathbf{q})$ is typically a non-linear scaling of $\mathbf{q}$.

3 Related Work

Numerous recent methods show how to learn dynamics via physically informed networks.

3.1 Functional Priors

Functional priors embed the full functional form of an equation into the neural network. Physics-informed Neural Networks (PINNs) [11, 12] and Hamiltonian Nets (Marios) look at directly embedding the equations into the loss function. PINNs furthermore exploit autograd to compute partial-derivatives and show that such an approach is more powerful than symbolic regression.

3.2 Integrator Priors

Many methods seek to use a NN to parametrize the time derivatives of a state vector and then use this learned parametrization in a scientific integrator such as a Runge-Kutta method. The authors of NeuralODE [2] 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. The paper reintroduces the adjoint method to ensure linear scaling of the gradient computations. Other work such as [19] theoretically proves that the embedded integrator

in Hamiltonian Neural Networks significantly benefits from being symplectic.

3.3 Graph based methods

In [1], the authors show 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 [15, 14, 3].

3.4 Lagrangian/Hamiltonian Methods

While [3] and [6] propose the explicit learning of Hamiltonian and Lagrangian functions, [9] 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.

3.5 Latest Advances

Constrained HNN [5], one of the most recent advances published as a spotlight in NeurIPS 2020, 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.

4 Method

The architecture for our method can be seen in figure 2. We learn 3 components: the Hamiltonian H_{θ_1}, the driving force F_{θ_2} and the damping term D_{θ_3} each parametrized by a neural network with weights θ_i. Using these components, we can define the predicted state time derivatives as in equation 4.

The equation takes the form of the Port-Hamiltonian described earlier with two modifications. First, our approach exploits the fact that most damped systems only consist of a non-zero, state-independent term in the lower right quadrant, so we replace $\mathbf{D}(\mathbf{q})$ with a single weight parameter independent of $\mathbf{q}$ in the lower right half of the matrix. Secondly, in order to generalize to time dependent forcing, we replace $g(\mathbf{q})u$ with $\mathbf{F}_{\theta_2}(\mathbf{t})$.

$$\begin{bmatrix}\dot{\mathbf{q}}\\ \dot{\mathbf{p}}\end{bmatrix} = \left(\begin{bmatrix}0 & \mathbf{I}\\ -\mathbf{I} & 0\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & \mathbf{D}_{\theta_2}\end{bmatrix}\right)\begin{bmatrix}\frac{\partial \mathcal{H}_{\theta_1}}{\partial \mathbf{q}}\\ \frac{\partial \mathcal{H}_{\theta_1}}{\partial \mathbf{p}}\end{bmatrix} + \begin{bmatrix}0\\ \mathbf{F}_{\theta_3}(\mathbf{t}))\end{bmatrix} \quad (4)$$

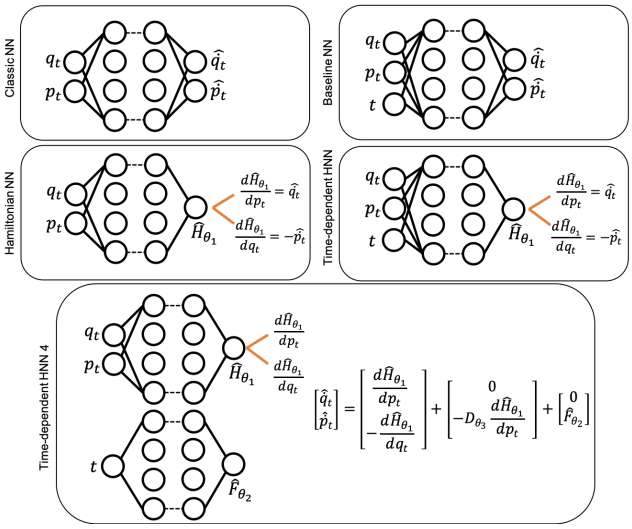


Figure 2: Architectures used to learn dynamics in this paper. The naive extension of classic NN and Hamiltonian NN (top left) is to incorporate time as an additional input variable (top right). Our innovation, which exploits Port-Hamiltonians, explicitly learns the force F_{θ_2} as well as the damping coefficient D_{θ_3}.

We define the loss function for our architecture as follows:

$$\mathcal{L} = \left\| \frac{\partial \mathbf{p}}{\partial t} + \frac{\partial \mathcal{H}_{\theta_1}}{\partial \mathbf{q}} + \mathbf{D}_{\theta_2}\frac{\partial \mathcal{H}_{\theta_1}}{\partial \mathbf{p}} + F_{\theta_3}(t)\right\|_2^2 + \left\| \frac{\partial \mathbf{q}}{\partial t} - \frac{\partial \mathcal{H}_{\theta_1}}{\partial \mathbf{p}}\right\|_2^2 + \alpha_{reg}||\mathbf{F}_{\theta_3}(\mathbf{t})||_1 + \beta_{reg}||\mathbf{D}_{\theta_2}||_1 \quad (5)$$

To learn the dynamics, we feed in a state-vector $[\mathbf{q}, \mathbf{p}, \mathbf{t}]$ to our model. The first component $\mathcal{H}_{\theta_1}$ consists of 3 hidden layers designed to predict $\mathcal{H}$ from $[\mathbf{q}, \mathbf{p}]$ data. The second component $\mathbf{D}_{\theta_3}$ consists of a single weight parameter (i.e. θ_2 is a single node) designed to learn the damping $\mathbf{D}$ and the third neural-network consists of 3 hidden layers designed to predict $\mathbf{F}(\mathbf{t})$. We use an L2-norm penalty for the predicted state-vectors and an L1-norm for force and damping to encourage sparsity. We introduce the sparsity-inducing L1-norm on force and damping because we wanted our network to identify both classical autonomous systems (which may not have force or time-varying components) as well as non-autonomous systems. For our experiments we use 200 nodes per layer and find that most activations, including tanh, sin and cos yield comparable results. To benchmark our method, we use a modified classic NN that takes in an additional time vector and predicts $[\dot{\mathbf{q}}, \dot{\mathbf{p}}]$. We also take the straightforward extension of HNN to include time as a variable input. Since all the networks are designed to output state vector derivatives with respect

to time, at inference, we use a Runge-Kutta 4th order integrator to integrate initial conditions from $t = 0$ to $t = T_{max}$ which we refer to as a state rollout.

5 Results

We benchmark the performance of our models on numerous datasets that canvas time-independent systems to complex chaotic forced systems.

5.1 Simple Mass Spring

The simple mass spring system is a well known system from classical physics that obeys Hooke's Law. The Hamiltonian for a simple mass spring system is written as:

$$\mathcal{H} = \frac{1}{2}k\mathbf{q}^2 + \frac{1}{2m}\mathbf{p}^2 \tag{6}$$

where k is the spring constant and m is the mass of the block.

Training: For simplicity, we set k and m to 1 for our experiments. We sample 25 initial training conditions that satisfy $q_0^2 + p_0^2 = r_0^2$ where $0.5 \leq r_0 \leq 1.5$ as is done in [6]. The trajectories are sampled with $\Delta t = 0.05$ and $T_{max} = 3.05$.

Test: At test time, we feed an initial condition and evolve it to $T_{max} = 9.1$ which is 3 times the training time. The test error in fig.3 is measured as the Mean Squared Error (MSE) between the ground truth and predicted state and energy values for a given trajectory. We also show the average State and Energy MSE across 50 trajectories.

We investigate this simple system to show that learning a separate, regularised forcing term results in much better state and energy predictions in comparison to TDHNN and baseline NN. The main reason for this is that while TDHNN and baseline NN learn dynamics with time-steps within the training regime, they typically cannot generalize to unseen time-steps. Learning a separate forcing term and regularizing it keeps the time component independent of the Hamiltonian and therefore allows us to nearly match the performance of classical time-implicit HNN.

5.2 Damped Mass Spring

Damping is generally not considered part of Hamiltonian systems. For example, let us take the following damped system:

$$\ddot{\mathbf{q}} = -\mathbf{q} - \delta\dot{\mathbf{q}} \tag{7}$$

and we know $\mathbf{p} = m\dot{\mathbf{q}}$ which implies $\dot{\mathbf{q}} = m^{-1}\mathbf{p}$.

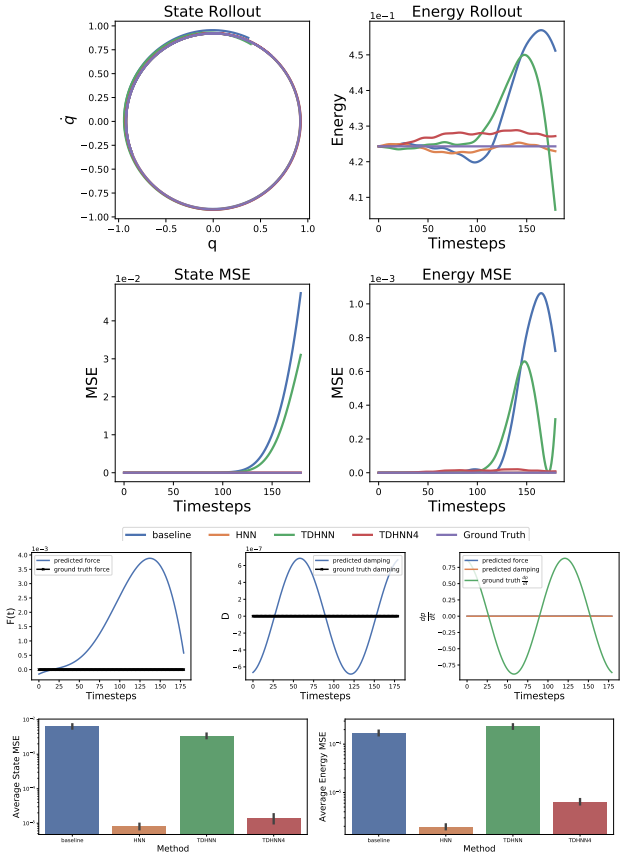


Figure 3: Simple Mass Spring system has no explicit time dependence. We see that TDHNN4 can almost recover the dynamics of HNN. The result is achieved by regularising the force and friction terms as can be seen in the figure. Baseline and TDHNN are unable to achieve the same test state error as they are only reliable for time steps that were within the training regime.

Then, the integral of the right hand side with respect to q will give us:

$$\frac{\mathbf{q}^2}{2} + \delta\mathbf{q}\dot{\mathbf{q}} \tag{8}$$

The equation above looks like a modified potential function which can be combined with a kinetic energy term to give a Hamiltonian as:

$$\mathcal{H} = \frac{\mathbf{p}^2}{2m} + \frac{\mathbf{q}^2}{2} + \delta\mathbf{q}\dot{\mathbf{q}} \tag{9}$$

However, although we can recover the differential equation for $\ddot{\mathbf{q}}$ by $-\frac{\partial\mathcal{H}}{\mathrm{d}\mathbf{q}} = -\mathbf{q} - \delta\dot{\mathbf{q}} = \ddot{\mathbf{q}}$, we violate the rule that $\dot{\mathbf{q}} = m^{-1}\mathbf{p}$ since $\frac{\partial\mathcal{H}}{\mathrm{d}\mathbf{p}} = \dot{\mathbf{q}} + \delta\mathbf{q} \neq \dot{\mathbf{q}}$.

However, as we saw with Port-Hamiltonians, there is a

straightforward way to account for the damping term without explicitly defining a full Hamiltonian. By combining a regular Hamiltonian with an additional term that learns the damping, we can disentangle the two terms for learning. Indeed, there are alternative methods to do this, including generalised Hamiltonian decomposition.

Training: We have 20 initial conditions, uniformly sampled in $[-1, 1]^2$. Each trajectory is evolved until $T_{max} = 30.1$ with a $\delta t = 0.1$. We fix $\delta = 0.3$.

Test: At inference, we compute the average rollout MSE of 25 trajectories sampled and evaluated the same way as training, and report the average state and energy rollout MSE in fig.4.

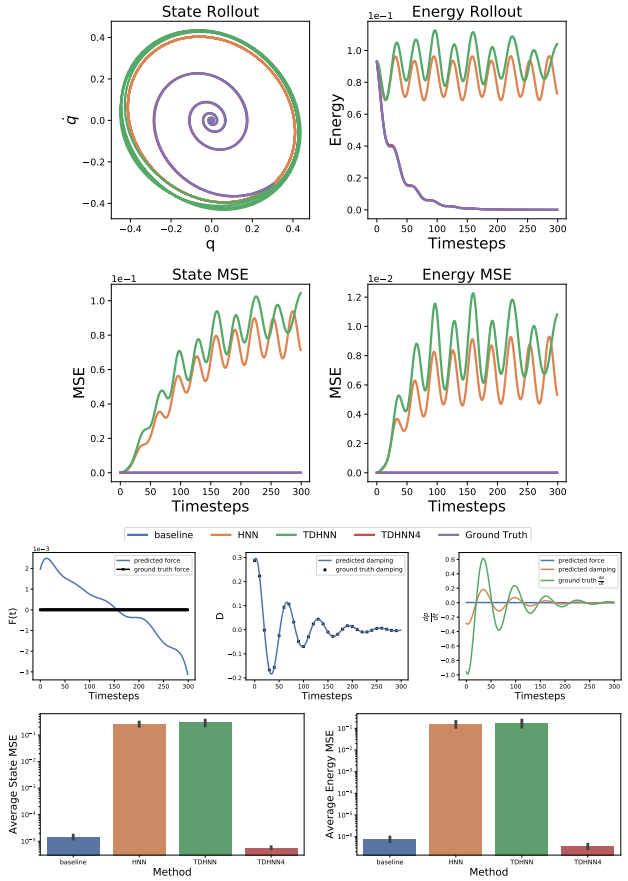


Figure 4: Damped mass spring setting: baseline and TDHNN4 recover the underlying dynamics. TDHNN4 is also able to perfectly learn the damping coefficient as predicted and ground truth friction are indistinguishable.

We can see that both baseline NN and TDHNN4 recover the dynamics well, whereas HNN and TDHNN struggle to learn the damping as there is no direct way of learning a Hamiltonian with damping.

5.3 Forced Mass Spring

Typically, while we cannot write the Hamiltonian for a damped system, we can write one for a forced system. Here, we study 2 types of forced mass-spring systems. The first has the following Hamiltonian form:

$$\mathcal{H} = \frac{1}{2}k\mathbf{q}^2 + \frac{1}{2m}\mathbf{p}^2 - \mathbf{q}\mathbf{F_0}sin(\omega\mathbf{t}) \tag{10}$$

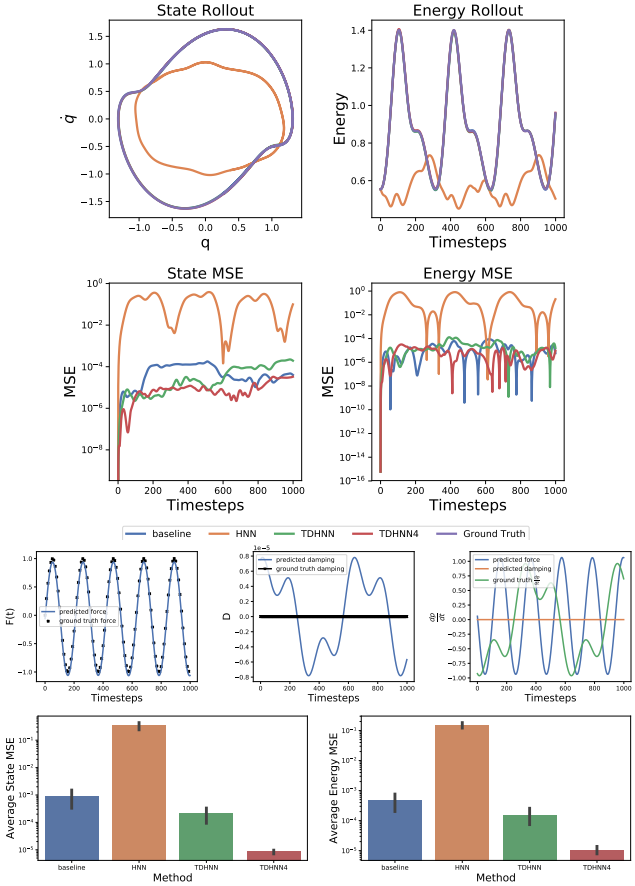


Figure 5: Forced mass spring setting: HNN cannot learn the underlying dynamics as it has no explicit-time dependence. TDHNN4 shows the best performance as it explicitly learns a time-dependent force.

The second has this Hamiltonian form:

$$\mathcal{H} = \frac{1}{2}k\mathbf{q}^2 + \frac{1}{2m}\mathbf{p}^2 - \mathbf{q}\mathbf{F_0}sin(\omega\mathbf{t})sin(2\omega\mathbf{t}) \tag{11}$$

Training: In both settings, we use 20 initial conditions, where the state is sampled from a radii between 0.5 and 1.5. The states are rolled out to $T_{max} = 10.01$ at a $\delta t = 0.01$. k and m are both set to 1. $\omega = 3$.

Test: At inference, we compute the rollout of 25 tra-

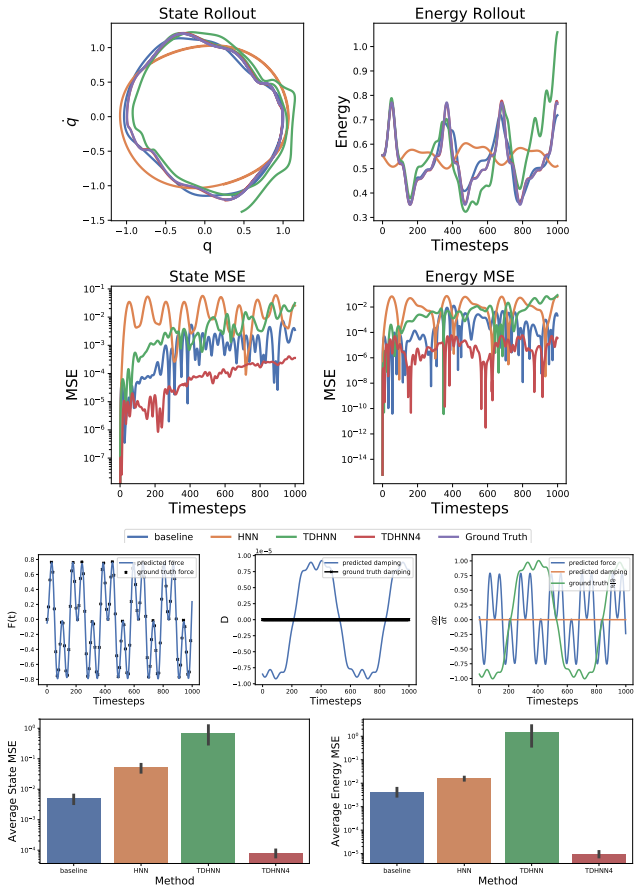


Figure 6: The time dependent force here is non-trivial, but TDHNN4 shows it can recover it precisely.

jectories, and report the average state and energy rollout MSE in the figures.

5.4 Duffing

The general duffing equation combines both the force and damping terms. Typically the equation is written as:

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

One can see that the unforced and undamped regular Hamiltonian of the duffing system is:

$$\mathcal{H}_{reg} = \frac{\mathbf{p}^2}{2m} + \alpha\frac{\mathbf{q}^2}{2} + \beta\frac{\mathbf{q}^4}{4} \tag{13}$$

This Hamiltonian therefore has a potential that can either be a single or a double well. A combination of parameters $\alpha, \beta, \delta, \gamma, \omega$ will make the duffing system either chaotic or non-chaotic. We study both regimes.

5.4.1 Non-Chaotic

Given a set of initial parameters where $\alpha = -1, \beta = 1, \delta = 0.3, \gamma = 0.2, \omega = 1.2$ 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 rollout 25 initial conditions at inference using the same conditions as above and evaluate both energy and state MSE.

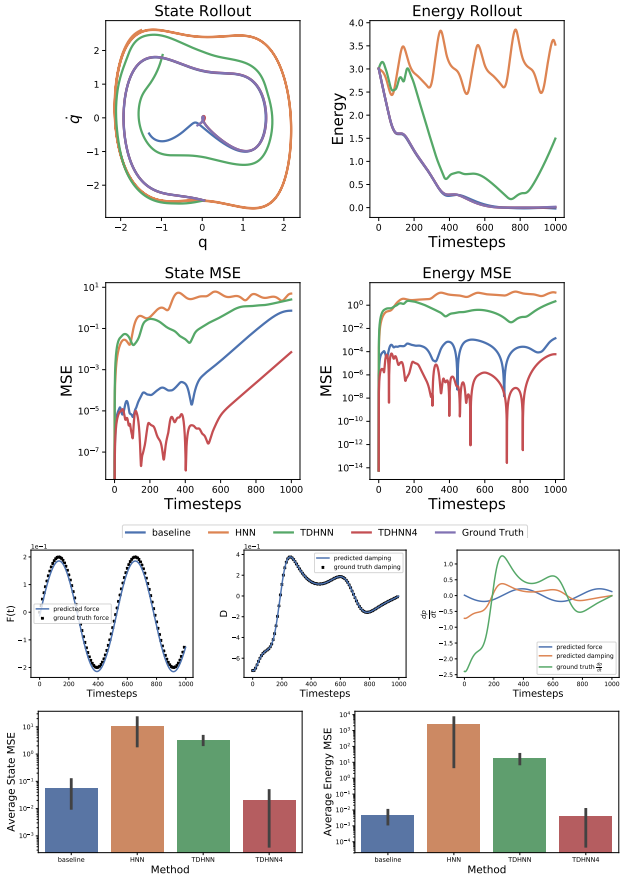


Figure 7: baseline NN and TDHNN4 both perform well in this setting. TDHNN4 is also able to extract the ground truth force and damping coefficient. The predicted Hamiltonian also visually appears closer to the ground truth in comparison to HNN or TDHNN.

5.4.2 Chaotic

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 the system on a single initial condition $[0, 0]$, rolled out to $T_{max} = 18,000$ with the same δt as training. One additional change we make at inference is a modification to the time variable which is fed in. 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 T. Our results are visually presented in Fig.1.

5.5 Relativity

We investigate the duffing equation with a relativistic kinetic energy term. The general Hamiltonian form is:

$$\mathcal{H} = c\sqrt{\mathbf{p}^2 + m^2c^2} + \frac{\alpha}{2}\mathbf{q}^2 + \frac{\beta}{4}\mathbf{q}^4 - \mathbf{q}\mathbf{F_0}sin(\omega \mathbf{t}) \quad (14)$$

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

Training: We train on 25 initial conditions, sampled in $[0, 3]^2$ uniformly. $T_{max} = 20.01$ and $\delta t = 0.01$. We set parameters s.t. $\alpha = 1, \beta = 1, \delta = 0, \gamma = 0.2, \omega = 1.2$.

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

6 Discussion

7 Conclusion

8 Questions/Thoughts

1. At present, baseline,HNN and TDHNN all use 3 layers with 200 weights per layer. TDHNN does the same to compute H, but it also has a separate network that learns F. Some could argue this is not fair - so how can we be 'fair' in our comparison?

2. The math does not checkout for the shifted force - should we keep trying to tune parameters so that force isn't offset? Note: even dissipative SympODEN learn a shifted force

3. we need to finalize the name for TDHNN4 as well as the title?

4. do we plan to arxiv this soon - to avoid the scoop?

5. does it suffice to investigate the mass spring system almost exclusively?

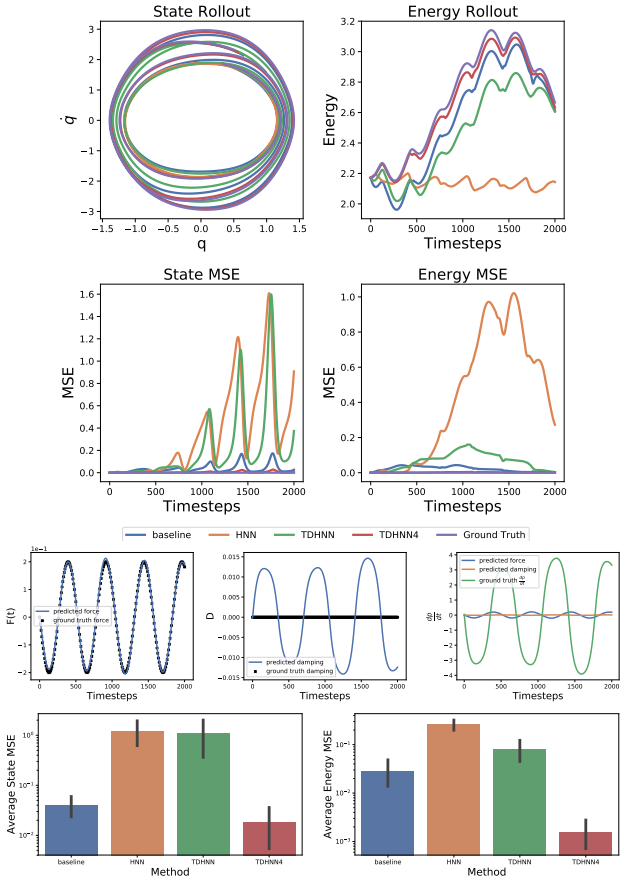


Figure 8: Learned dynamics of a relativistic duffing system. TDHNN4 recovers a shifted force with the right scaling and the H_{reg} component that illustrates a squashed phase space visible when relativistic kinetic energy is used.

6. can we explain why baseline does surprisingly well with damping and duffing? Somehow dissipation seems to help the baseline.

Table 1: Test Rollout MSE

Method	Ideal Mass Spring		Damped Mass Spring		Forced Mass Spring (1)		Forced Mass Spring (2)		Duffing Non-Chaotic	
	State	Energy	State	Energy	State	Energy	State	Energy	State	Energy
Baseline NN										
HNN										
TDHNN										
TDHNN4										

References

[1] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu. Interaction Networks for Learning about Objects, Relations and Physics. *arXiv:1612.00222 [cs]*, Dec. 2016. arXiv: 1612.00222.

[2] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018.

[3] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian Neural Networks. *arXiv:2003.04630 [physics, stat]*, Mar. 2020. arXiv: 2003.04630.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. arXiv: 1810.04805.

[5] M. Finzi, S. Stanton, P. Izmailov, and A. G. Wilson. Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data. *arXiv:2002.12880 [cs, stat]*, May 2020. arXiv: 2002.12880.

[6] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian Neural Networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15379–15389. Curran Associates, Inc., 2019.

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *arXiv:1703.06870 [cs]*, Jan. 2018. arXiv: 1703.06870.

[8] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, Jan. 1989.

[9] M. Lutter, C. Ritter, and J. Peters. Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning. *arXiv:1907.04490 [cs, eess, stat]*, July 2019. arXiv: 1907.04490.

[10] A. Pukrittayakamee, M. Malshe, M. Hagan, L. M. Raff, R. Narulkar, S. Bukkapatnum, and R. Komanduri. Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks. *The Journal of Chemical Physics*, 130(13):134101, Apr. 2009.

[11] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv:1711.10561 [cs, math, stat]*, Nov. 2017. arXiv: 1711.10561.

[12] 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. *Journal of Computational Physics*, 378:686–707, Feb. 2019.

[13] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia. Hamiltonian Graph Networks with ODE Integrators. *arXiv:1909.12790 [physics]*, Sept. 2019. arXiv: 1909.12790.

[14] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to Simulate Complex Physics with Graph Networks. *arXiv:2002.09405 [physics, stat]*, Feb. 2020. arXiv: 2002.09405.

[15] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv:1806.01242 [cs, stat]*, June 2018. arXiv: 1806.01242.

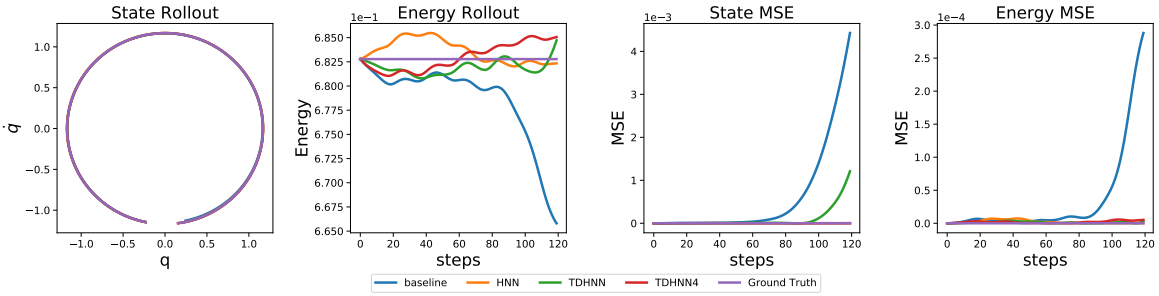
[16] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018.

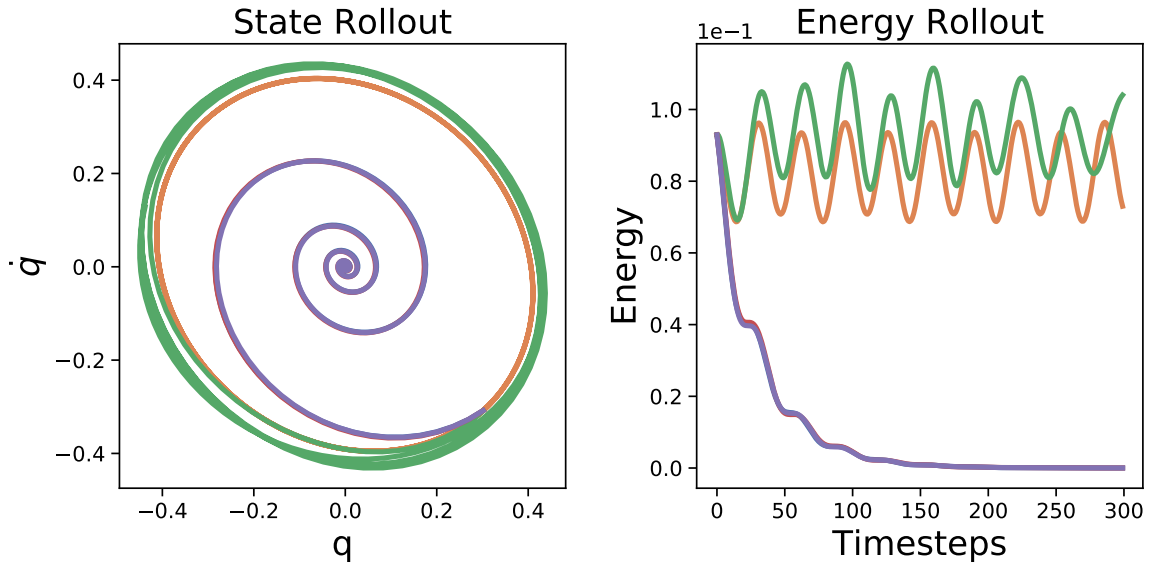
[17] K. Yao, J. E. Herr, D. Toth, R. Mckintyre, and J. Parkhill. The TensorMol-0.1 model chemistry: a neural network augmented with long-range physics. *Chemical Science*, 9(8):2261–2269, 2018.

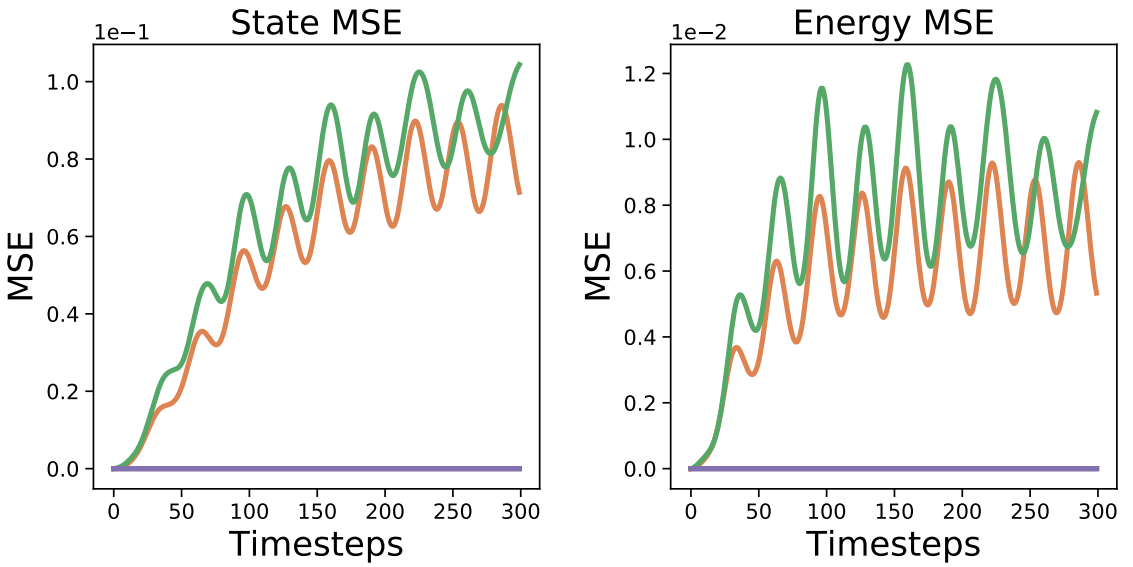
[18] Y. D. Zhong, B. Dey, and A. Chakraborty. Dissipative SymODEN: Encoding Hamiltonian Dynamics with Dissipation and Control into Deep

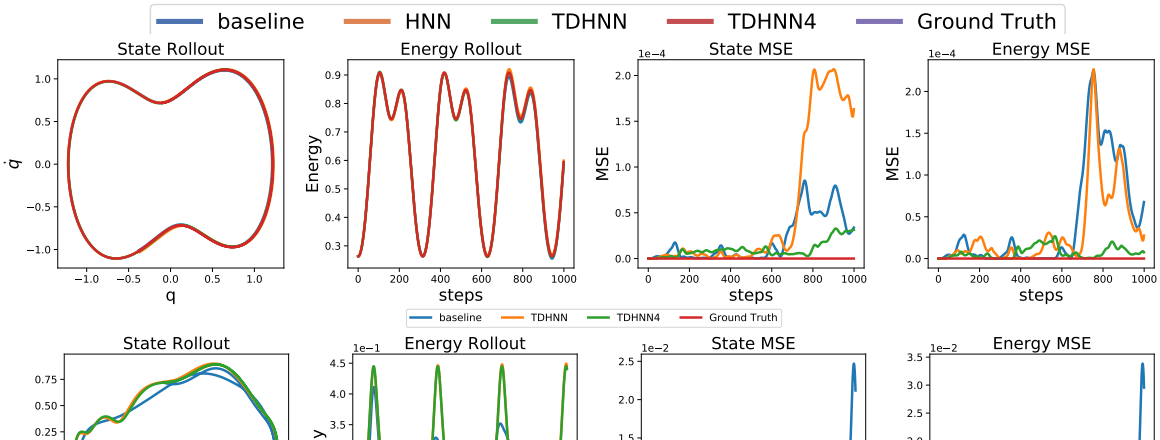
Learning. *arXiv:2002.08860 [cs, eess, stat]*, Apr. 2020. arXiv: 2002.08860.

[19] A. Zhu, P. Jin, and Y. Tang. Deep Hamiltonian networks based on symplectic integrators. *arXiv:2004.13830 [cs, math]*, Apr. 2020. arXiv: 2004.13830.


State Rollout
Energy Rollout
State MSE
Energy MSE
baseline
HNN
TDHNN
TDHNN4
Ground Truth


State Rollout
Energy Rollout
Timesteps


State MSE
Energy MSE
Timesteps


baseline
HNN
TDHNN
TDHNN4
Ground Truth
State Rollout
Energy Rollout
State MSE
Energy MSE
steps