Dissipative Hamiltonian Neural Networks: Learning Dissipative and Conservative Dynamics Separately

Anonymous authors
Paper under double-blind review

Abstract

Understanding natural symmetries is key to making sense of our complex and ever-changing world. Recent work has shown that neural networks can learn such symmetries directly from data using Hamiltonian Neural Networks (HNNs). But HNNs struggle when trained on datasets where energy is not conserved. In this paper, we ask whether it is possible to identify and decompose conservative and dissipative dynamics simultaneously. We propose Dissipative Hamiltonian Neural Networks (D-HNNs), which parameterize both a Hamiltonian and a Rayleigh dissipation function. Taken together, they represent an implicit Helmholtz decomposition which can separate dissipative effects such as friction from symmetries such as conservation of energy. We train our model to decompose a damped mass-spring system. Then we apply it to two real-world datasets including a large ocean current dataset where decomposing the velocity field yields scientific insights.

1 Introduction and Related work

We are immersed in a complex, dynamic world where change is the only constant. And yet there are certain patterns to this change that suggest natural laws. These laws include conservation of mass, energy, and momentum. Taken together, they constitute a powerful simplifying constraint on reality. Indeed, physics tells us that a small set of laws and their associated invariances are at the heart of all natural phenomena. Whether we are studying weather, ocean currents, earthquakes, or molecular interactions, we should take care to respect these laws. And when we apply learning algorithms to these domains, we should ensure that they, too, take these laws into account.

One of the remarkable insights of physics is that all of these laws can be expressed as conserved quantities. For example, conservation of angular momentum enforces rotational invariance and conservation of mass-energy enforces time-translation invariance (Noether, 1918). The mathematical framework for representing these conserved quantities is called Hamiltonian mechanics, and the function which relates the coordinates of a physical system to a conserved quantity (usually energy) is called the Hamiltonian. Recent work by (Greydanus et al., 2019), (Toth et al., 2019) and others has shown that this function can be used to incorporate physical constraints, like conservation of energy, into deep neural networks.

These models, which we will refer to as Hamiltonian Neural Networks (HNNs), are able to learn more accurate representations of the world than would otherwise be possible. Recent works by (Cranmer et al., 2020), (Chen et al., 2019), and (Choudhary et al., 2020) have shown how to learn Hamiltonians for nonlinear, chaotic systems such as double pendulums, n-body systems, and bouncing balls. Moreover, these models can be learned starting from perceptually difficult observations, such as pixel videos. A number of other works have begun applying these physics-inspired models to new fields such as quantum mechanics (Broughton et al., 2020), chaos theory (Choudhary et al., 2020), and differential equations (Mattheakis et al., 2020; Desai et al., 2021).

One remaining drawback of HNNs is that they assume that the total energy of the observed system is conserved. In other words, they assume the absence of friction and other sources or sinks of energy. This obscures an important fact about the real world: no matter how sensitive one's measurements are, real-world data never conforms exactly to theoretical expectations. Real-world dynamic systems

often contain dissipative dynamics due to friction, external forces, or measurement noise. How might we enable HNNs to learn conserved quantities even in the presence of noisy real-world data?
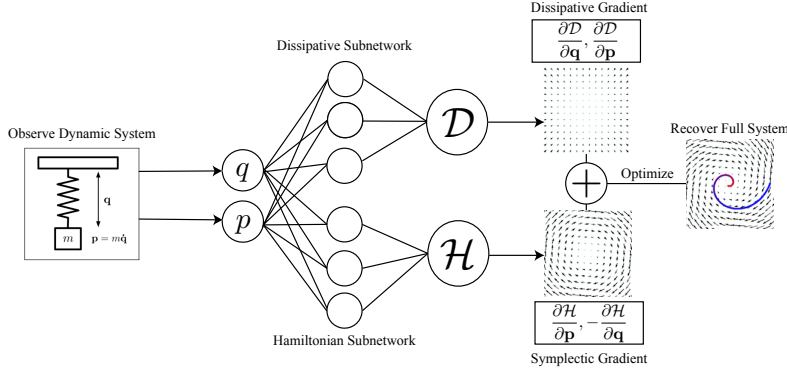

Figure 1: Visualizing a D-HNN. D-HNNs leverage two neural networks to model dynamic systems. Like an HNN, we parameterize a Hamiltonian function $\mathcal{H}$ with the first subnetwork and use it to model part of the system's dynamics (lower part of diagram). Unlike an HNN, we use a second subnetwork to allow our parametric model to output a second scalar function, $\mathcal{D}$, and use it to obtain an ordinary, non-symplectic gradient which can be used to model dissipative dynamics. The sum of these two fields can be used to model dynamic systems which have both conservative and dissipative dynamics: in this case, a damped spring.

We tackle the problem of learning in real-world dynamic systems by introducing Dissipative Hamiltonian Neural Networks (D-HNNs), a model that extends HNNs to learn the decomposition of any dynamic system into its dissipative and conserved quantities. D-HNNs learn decompositions by learning a Rayleigh dissipation function in addition to a Hamiltonian. The resulting network has two scalar outputs, $\mathcal{H}$ and $\mathcal{D}$. The first yields a rotational vector field with respect to the input coordinates and the second yields an irrotational vector field over the same coordinates. Summed together, these two vector fields can fit any arbitrary vector field (see discussion of Helmholtz-Hodge decomposition in appendix). When we train our model on a dataset, it learns an implicit Helmholtz decomposition of the dynamics. In other words, it learns the dissipative and conservative dynamics separately. When we train it on observations of a damped spring, for example, our model learns a Hamiltonian resembling that of a non-damped spring while using the Rayleigh dissipation function to model the effects of friction.

Learning decompositions of dynamic systems has practical applications for other fields of vector analysis such as computer graphics, fluid dynamics, weather modeling, and even forensics (Bhatia et al., 2012). One example we consider in this work is oceanography. Oceanographers are often interested in locating upwelling, a dissipative feature, which occurs when cold, nutrient-rich waters rise to the surface of the ocean, supporting large populations of fish and other marine life. Finding such dissipative patterns in ocean current data is a non-trivial task. In this work, we apply D-HNNs to a large ocean-current dataset and use them to map such features.

2 METHODS

D-HNNs perform an implicit Helmholtz decomposition on a dynamic system's vector field V that describes all the possible states of the system—and extract V_{irr} and V_{rot}, which correspond to the dissipative and conserved quantities of the system, respectively. D-HNNs are parameterized as a neural network, with parameters θ, that leverage two subnetworks to learn two differentiable scalar functions, a dissipative function $\mathcal{D}(\mathbf{q}, \mathbf{p})$ and a Hamiltonian function $\mathcal{H}(\mathbf{q}, \mathbf{p})$. During the forward pass, each subnetwork digests input canonical coordinates (here $\mathbf{q}, \mathbf{p} \in \mathbb{R}^n$) that describe the state of the system and then outputs scalar values for $\mathcal{D}$ and $\mathcal{H}$. Next, we take the in-graph gradient of both scalars with respect to the inputs. Finally, we compute an L2 loss between the sum of these input gradients and the target gradients, and backpropagate error to all trainable parameters. Equation 1

gives the particular form of our loss function.

$$L_{DHNN} = \left\| (\frac{\partial \mathcal{H}}{\partial \mathbf{p}} + \frac{\partial \mathcal{D}}{\partial \mathbf{q}}) - \frac{\partial \mathbf{q}}{\partial t} \right\|_2 + \left\| (-\frac{\partial \mathcal{H}}{\partial \mathbf{q}} + \frac{\partial \mathcal{D}}{\partial \mathbf{p}}) - \frac{\partial \mathbf{p}}{\partial t} \right\|_2 \quad (1)$$

Intuitively, we begin by summing the symplectic gradient of $\mathcal{H}$ and the regular gradient of $\mathcal{D}$. This quantity should match the time derivatives of the input coordinates after training. In order to make this happen, we compute an L2 loss between it and the true time derivatives of the system. Figure 1 provides a visual summary of this process.

Baselines. We compare the performance of D-HNNs to HNNs, a baseline multi-layer perceptron (MLP) model, and a numerical decomposition method because they have the most relevance to our performance metrics.

Training. We trained three models on the dataset: a D-HNN, an HNN, and an MLP that directly mapped input coordinates $(\mathbf{q}, \mathbf{p})$ directly to $(\frac{\partial \mathbf{q}}{\partial t}, \frac{\partial \mathbf{p}}{\partial t})$. The HNN and MLP models contained two layers of 256 neurons. Each of the D-HNN's subnetworks contained one layer of 256 neurons and a residual layer. We shuffled the dataset prior to training to eliminate order effects. We backpropogated our model using Equation 1 as our loss function. We synthetically constructed the dataset and analytically calculated the targets using Equation 2 and 6. We used the analytical time derivatives of the system as the targets. On each task, we trained each model with a learning rate of 10^{-2} for 5000 gradient steps, using a 80/20 train/test percentage split (see Appendix for more on hyperparameters). We used the Adam optimizer Kingma & Ba (2014) and conducted batch gradient descent with a batch size of 128. We implemented our models in PyTorch and trained each model with P100 GPUs.

Metrics. At inference time, we report the mean-squared error (MSE) between the true coordinates and the target coordinates. To obtain the reported dynamics and energy metrics, we used an initial state starting at $(0.9, 0)$ and used the fifth-order Runge-Kutta integrator in `scipy.integrate.solve_ivp` with the error tolerance set to 10^{-12}. We also compared our models' decomposition performance to the Gauss-Seidel method, an iterative numerical decomposition that solves Poisson's Equation (for more details, see Stam (2003)).

Table 1: Comparing D-HNN training and evaluation metrics to a number of baseline approaches. The test MSE metric measures the difference between the predicted time derivatives and true time derivatives of the system, and the energy MSE metric measures the differences between the predicted energies and true energies. The D-HNN outperforms the MLP and HNN on all tasks, though sometimes only by a small margin. The MLP is slightly behind the D-HNN in predicting energies and state trajectories. We also tried running a numerical Helmholtz decomposition on the mass-spring dataset (using the Gauss-Seidel method to solve Poisson's equation). This introduced a mean square error of $7.3 \cdot 10^{-3}$ which was mostly due to interpolation error.

Metric	Model	Damped spring	Real pendulum	OSCAR ocean data
Test MSE	MLP	$9.9 \cdot 10^{-5}$	$1.1 \cdot 10^{-3}$	0.181
	HNN	$6.3 \cdot 10^{-1}$	$1.4 \cdot 10^{-3}$	0.146
	D-HNN	$\mathbf{5.8 \cdot 10^{-5}}$	$\mathbf{9.5 \cdot 10^{-4}}$	**0.143**
Energy MSE	MLP	$6.0 \cdot 10^{-6}$	$1.2 \cdot 10^{-3}$	–
	HNN	$2.9 \cdot 10^{-1}$	$1.0 \cdot 10^{-2}$	–
	D-HNN	$\mathbf{3.6 \cdot 10^{-6}}$	$\mathbf{1.1 \cdot 10^{-3}}$	–

3 MODEL TASKS AND RESULTS

Task 1: Damped spring. A damped mass-spring system can be described as simple harmonic oscillator with a velocity-dependent friction force. The Hamiltonian and Rayleigh functions of such a spring can be described by Equation 2 and 3 respectively.

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

$$\mathcal{D} = \frac{1}{2}\rho\dot{q}^2 \quad ; \quad F_i^f = -\frac{\partial\mathcal{D}(\dot{\mathbf{q}})}{\partial\dot{\mathbf{q}}} \tag{3}$$

Here k is the spring constant, q is the distance from the origin, ρ is the friction coefficient, p is the momentum variable and m is the mass constant. For simplicity, we set $k = m = 1$. We initially set $\rho = 2$, but $\rho = 1, 4$ when making counterfactual adjustments (Figure 4 to the friction coefficient.

Figure 6 shows the results of training the baseline model (MLP), a HNN and a D-HNN on a damped mass-spring dataset. The D-HNN constructed an exact decomposition of the system, extracted its conservative and dissipative quantities, and modeled its dynamics with high-accuracy. It outperformed both the HNN and MLP as well as the Gauss-Seidel numerical decomposition. The MLP gives a good fit and can be integrated as a Neural ODE Chen et al. (2018), but it cannot reproduce a decomposition of the the dissipative and conservative quantities of the system. The HNN was only able to extract the conserved quantity of the mass-spring system when trained with the same dataset. The HNN yielded the worst fit of the three models because it incorrectly assumed that energy is strictly conserved.

Because D-HNNs learn exact decompositions, they can recover the dynamics of the full system, which is simply the sum of the dissipative and conserved dynamics. These decompositions even permit new forms of generalization. For example, we can multiply the dissipative component by various coefficients in order to answer the counterfactual question, *what would the dynamics look like if we halved or doubled the friction coefficient?* Figure 4 does this with the damped spring model; we see that the D-HNN's trajectories are close to ground truth even though it was never exposed to different friction coefficients during training. We emphasize that other parametric models such as MLPs are unable to perform this kind of generalization.
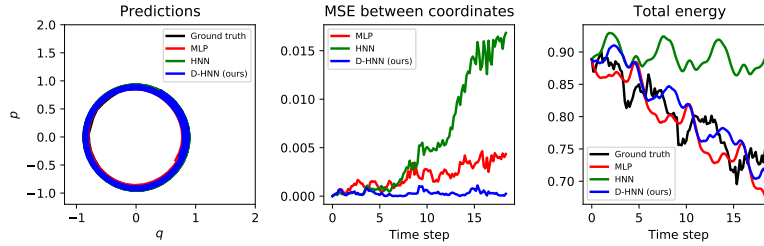

Figure 2: Analyzing a model trained on data from a real, slightly damped pendulum. **Left:** Predicted dynamics. The D-HNN trajectory covers the MLP and HNN trajectories. The D-HNN recovers the system with high accuracy and obscures the ground truth dynamics. **Middle:** The mean square error (MSE) between predicted coordinates and the true coordinates. D-HNNs outperform the other models. **Right:** The predicted energies of the pendulum system from each model compared to the ground truth. The D-HNN, which properly models both conservative and dissipative forces, slightly outperforms the MLP.

Task 2: Real Damped Pendulum System. A damped pendulum system can be described as a nonlinear harmonic oscillator inhibited by a friction force. The Hamiltonian and Rayleigh functions of such a system can be respectively described by Equation 3 and 4 respectively.

$$\mathcal{H} = 2mgl(1 - \cos q) + \frac{l^2 p^2}{2m} \tag{4}$$

For simplicity, we set $l = m = 1$. However, we set our gravity constant $g = 3$. We perform the same training procedures and metrics reporting that were done on Task 1, with three exceptions. First, we feature position and momentum readings of a real pendulum with friction from Schmidt & Lipson (2009). Second, we do not use the Gauss-Seidel decomposition as a comparison because the dataset already consists of trajectories, and thus we do not need to use an analytic method to estimate a trajectory. Third, we report the MSE of the total and predicted energy by integrating along the trajectory from the first point in the test set.

Figure 5 shows that the D-HNN improves over the baseline model and the HNN in terms of both mean square error between coordinate trajectories and mean square error between predicted and

actual total energy. The baseline MLP, meanwhile, does a slightly worse job of predicting a trajectory and conserving total energy. However, it still improves over the HNN. The HNN accrues the most error out of the three models, by a factor of ten in both trajectory and energy MSE. We observe this effect because HNNs explicitly force energy to be conserved in the system, which is a poor assumption for this dataset, which contains some friction.

Task 3: Surface currents. We tested D-HNNs on decomposing velocity fields of surface currents in the Atlantic Ocean. We use the OSCAR dataset from Earth and Space Research. For our task, we extracted 69 time frames, each consisting of a 50x50 grid of surface currents from the Atlantic Ocean. The ways of the ocean are complex. Oceanographers are typically interested in modeling ocean flow and identifying features such as eddy currents, upwelling and downwelling (Zhang et al., 2019; Chelton et al., 2011; Petersen et al., 2013). Upwelling is a dissipative feature that occurs when cold, nutrient-rich waters rise to the surface of the ocean. Conversely, downwelling is an opposite process by which surface water descends. These features tend to occur in the centers of vortices, which form in order to preserve the geostrophic balance. Oceanographers often describe current velocity fields by composing a divergence-free stream function ψ (conserved) and some curl-less velocity potential ϕ (dissipative) (Buhler et al., 2014; Caballero et al., 2020).

Our experiment showed that D-HNNs can separately model ψ and ϕ—both unknown *a priori*—and uncover ocean flow features by performing an implicit Helmholtz decomposition across 69 time frames. From the bright and dark regions seen in the H's learned scalar fields (Figure 3, we can identify eddies by locating the centers of vortices. To maintain geostrophic balance in a vortex, water will move upwards or downwards depending on the direction of the vortex. By identifying the centers of vortices, we can find where upwelling and downwelling features will be the strongest.
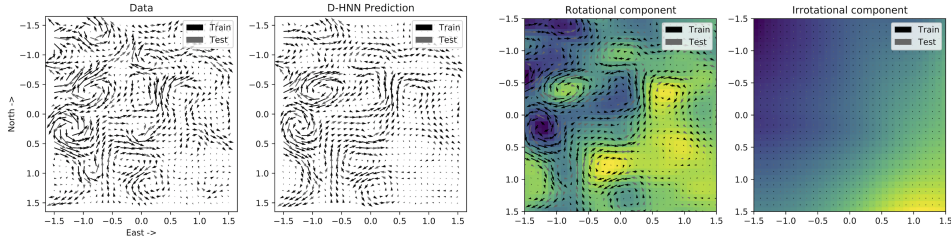

Figure 3: A learned decomposition of surface current velocities from the OSCAR ocean dataset. Backgrounds in plots 3 and 4 are colored according to scalar fields $\mathcal{H}$ and $\mathcal{D}$ respectively. In Plot 3, the regions which are particularly blue or yellow correspond to centers of vortices. These regions tend to be associated with upwelling and downwelling and are of ecological importance.

D-HNNs only performed decompositions slightly better than HNNs because ocean flow is typically much more conservative than dissipative and thus makes modeling an easier task for HNNs. Eddies have a much more prominent effect on the velocity fields than the dissipative features of upwelling and downwelling (Figure 3). We find that upwelling and downwelling does occur within the examined region but with a much fainter effect on the velocity field than eddies. This same effect is observed in (Zhang et al., 2019) where the authors deploy non-ML Helmholtz decompositions for eddy detection.

4 DISCUSSION

D-HNNs build on the foundations of HNNs by bridging the gap between the clean, natural symmetries of physics and the imperfect, messy data often found in the real world. They employ the tools of Hamiltonian mechanics and Helmholtz decomposition to separate conserved quantities from dissipative quantities. In doing so, they allow us to model complex physical systems while enforcing strict conservation laws. They represent a small, practical advance in that they can help us see our datasets, like the ocean currents dataset, in a new way. And at the same time, they represent progress towards the more ambitious goal of building models which make sense of the incredible complexity of the real world by focusing on its invariant quantities.

REFERENCES

M Akram and V Michel. Regularisation of the helmholtz decomposition and its application to geomagnetic field modelling. *GEM-International Journal on Geomathematics*, 1(1):101–120, 2010.

John Bell, Louis Howell, and Phillip Colella. An efficient second-order projection method for viscous incompressible flow. In *10th Computational Fluid Dynamics Conference*, pp. 1560, 1991.

Harsh Bhatia, Gregory Norgard, Valerio Pascucci, and Peer-Timo Bremer. The helmholtz-hodge decomposition—a survey. *IEEE Transactions on visualization and computer graphics*, 19(8): 1386–1404, 2012.

Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Murphy Yuezhen Niu, Ramin Halavati, Evan Peters, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

Oliver Buhler, Jorn Callies, and Raffaele Ferrari. Wave–vortex decomposition of one-dimensional ship-track data. 2014.

Ainhoa Caballero, Sandrine Mulet, Nadia Ayoub, Ivan Manso-Narvarte, Xabier Davila, Christine Boone, Florence Toublanc, and Anna Rubio. Integration of hf radar observations for an enhanced coastal mean dynamic topography. *Frontiers in Marine Science*, 7:1005, 2020.

Dudley B Chelton, Michael G Schlax, and Roger M Samelson. Global observations of nonlinear mesoscale eddies. *Progress in oceanography*, 91(2):167–216, 2011.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. *8th International Conference on Learning Representations (ICLR 2020)*, 2019.

Anshul Choudhary, John F Lindner, Elliott G Holliday, Scott T Miller, Sudeshna Sinha, and William L Ditto. Physics-enhanced neural networks learn order and chaos. *Physical Review E*, 101(6):062207, 2020.

Douglas Cline. *Variational principles in classical mechanics*. University of Rochester River Campus Librarie, 2017.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

Shaan A Desai, Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen J Roberts. Port-hamiltonian neural networks for learning explicit time-dependent dynamical systems. *Physical Review E*, 104(3):034312, 2021.

Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *arXiv preprint arXiv:1906.01563*, 2019.

David J Griffiths. Introduction to electrodynamics, 2005.

H von Helmholtz. Über integrale der hydrodynamischen gleichungen, welche den wirbelbewegungen entsprechen. *Journal für die reine und angewandte Mathematik*, 1858(55):25–55, 1858.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

Marios Mattheakis, David Sondak, Akshunna S Dogra, and Pavlos Protopapas. Hamiltonian neural networks for solving differential equations. *arXiv preprint arXiv:2001.11107*, 2020.

E Minguzzi. Rayleigh's dissipation function at work. *European Journal of Physics*, 36(3):035014, 2015.

Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. Physically based modeling and animation of fire. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 721–728, 2002.

Emmy Noether. Invariant variation problems. *Transport Theory and Statistical Physics*, 1(3):186–207, 1918.

Biswaroop Palit, Anup Basu, and Mrinal K Mandal. Applications of the discrete hodge helmholtz decomposition to image and video processing. In *International Conference on Pattern Recognition and Machine Intelligence*, pp. 497–502. Springer, 2005.

Mark R Petersen, Sean J Williams, Mathew E Maltrud, Matthew W Hecht, and Bernd Hamann. A three-dimensional eddy census of a high-resolution global ocean simulation. *Journal of Geophysical Research: Oceans*, 118(4):1759–1774, 2013.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

Jos Stam. Real-time fluid dynamics for games. In *Proceedings of the game developer conference*, volume 18, pp. 25, 2003.

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. *arXiv preprint arXiv:1909.13789*, 2019.

Cuicui Zhang, Hao Wei, Chongke Bi, and Zhilei Liu. Helmholtz–hodge decomposition-based 2d and 3d ocean surface current visualization for mesoscale eddy detection. *Journal of Visualization*, 22 (2):231–243, 2019.

A THEORY

D-HNNs learn to decompose the dynamics of physical systems into their dissipative and conservative components. To do this, they employ the tools of Hamiltonian mechanics and the Helmholtz decomposition.

A.1 HAMILTONIAN MECHANICS

We begin by providing a brief overview of Hamiltonian mechanics, an essential tool for modeling the time evolution of dynamic systems. The Hamiltonian $\mathcal{H}(\mathbf{q}, \mathbf{p})$ is scalar function such that

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}} = \frac{\partial \mathbf{q}}{dt}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} = \frac{\partial \mathbf{p}}{dt} \tag{5}$$

where $(\mathbf{q}, \mathbf{p})$, composed of pairs $(q_1, p_1)...(q_N, p_N)$, are the coordinates of the system. Moving in the direction of the *symplectic* gradient $S_{\mathcal{H}} = (\frac{\partial \mathcal{H}}{\partial \mathbf{p}}, -\frac{\partial \mathcal{H}}{\partial \mathbf{q}})$ keeps the value of $\mathcal{H}$'s output constant throughout the time evolution of the system, thereby representing a time-independent conserved quantity. Using $\mathcal{H}$, we can model the exact conserved quantities of both simple and complex dynamic systems.

Nonconservative systems. Real dynamic systems often contain dissipative forces such as friction or radiation, and these forces cause energy loss over time. Thus energy is not strictly conserved, violating the assumptions of regular Hamiltonian mechanics. Therefore we must work within the framework of nonconservative systems when predicting dynamics in dissipative environments. The Rayleigh dissipation function $\mathcal{D}(\dot{\mathbf{q}})$ is a scalar function that provides a way to account for generalized, velocity-dependent dissipative forces—such as friction—in the context of Hamiltonian mechanics Cline (2017). Equation 6 shows the Rayleigh function for linear, velocity-dependent dissipation. Here ρ is a constant and $\dot{q}$ is the velocity coordinate.

$$\mathcal{D} = \frac{1}{2}\rho\dot{q}^2 \tag{6}$$

Equation 7 shows that nonlinear, velocity-dependent dissipation forces F_i within the system can be accounted for as long as they are defined as

$$F_i^f = -\frac{\partial \mathcal{D}(\dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}} \tag{7}$$

In-depth applications of the Rayleigh dissipation function can be found in Minguzzi (2015).

A.1.1 HELMHOLTZ DECOMPOSITION

Like many students today, Hermann von Helmholtz realized that medicine was not his true calling. Luckily for us, he pursued physics instead and discovered one of the most useful tools in vector analysis: the Helmholtz decomposition. Helmholtz's theorem states that any smooth vector field V can be expressed as a sum of its irrotational component V_{irr} and its rotational component V_{rot} Helmholtz (1858). More formally, V can be expressed as the sum of the curl-less gradient of a scalar potential ϕ and the divergence-less curl of a vector potential $\mathbf{A}$.

$$V = V_{irr} + V_{rot} = \nabla\phi + \nabla \times \mathbf{A} \tag{8}$$

D-HNNs performing an implicit Helmholtz decomposition can be found in Figure 6 and Figure 4.

Applications. Scientists have leveraged the Helmholtz decomposition to uncover features of interest in their data or to model different phenomena, most commonly different types of fluids. For example, the Helmholtz decomposition has been used to match fingerprints Palit et al. (2005), model geomagnetic fields Akram & Michel (2010), and accurately animate fire Nguyen et al. (2002).

Approximations. The Helmholtz decomposition can be implemented with approximate numerical methods such as the least squares finite-element method Bhatia et al. (2012); Stam (2003); Bell et al. (1991). While the decomposition of a vector field is uniquely determined with appropriate boundary conditions Griffiths (2005), numerical decompositions are not: just as there are numerous ways for a model to fit data, there are numerous ways to decompose a vector field. In this work, for example, we

follow Stam (2003), using the Gauss-Seidel method to obtain baseline numerical decompositions. One problem with most of these methods is that they are iterative and not easily parallelizable on GPUs. When dealing with a large dataset, the *entire* dataset typically needs to be considered, which can be computationally expensive and memory-intensive.

There is an even more insidious source of error: the grid-based interpolation one must do prior to decomposition. In our experiments, for example, the error due to imperfect interpolation was many orders of magnitude larger than the error due to numerical decomposition. In contrast, neural networks excel at interpolating data Hornik et al. (1989). We suspect that this is one reason they were able to improve over baseline approaches.
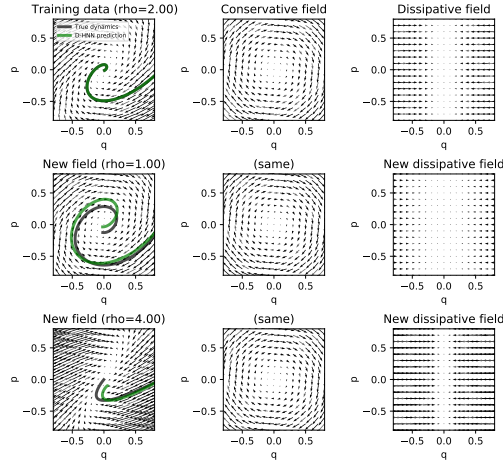
B FIGURES

Figure 4: Adjusting the dissipative component of a trained D-HNN so as to predict trajectories for *unseen* friction coefficients. Adjustments to the coefficients can be made by multiplying the learned Rayleigh dissipation function by various constants.
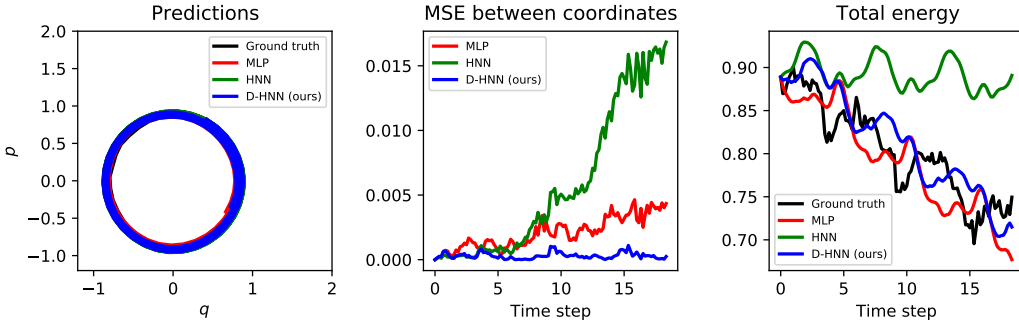

Figure 5: Analyzing a model trained on data from a real, slightly damped pendulum. **Left:** The predicted dynamics from all the models compared to the ground truth. The D-HNN trajectory covers up the MLP and HNN trajectories. The D-HNN recovers the system with high accuracy and obscures the ground truth dynamics. See Figure 6 for individual trajectories. **Middle:** The mean square error (MSE) between predicted coordinates and the true coordinates. D-HNNs clearly outperform the other models in providing the best accuracy along a given trajectory. **Right:** The predicted energies of the pendulum system from each model compared to the ground truth. Here, the D-HNN slightly outperforms the MLP. Thus the D-HNN excels at both predicting the right trajectories and modeling the system's dissipation over time.
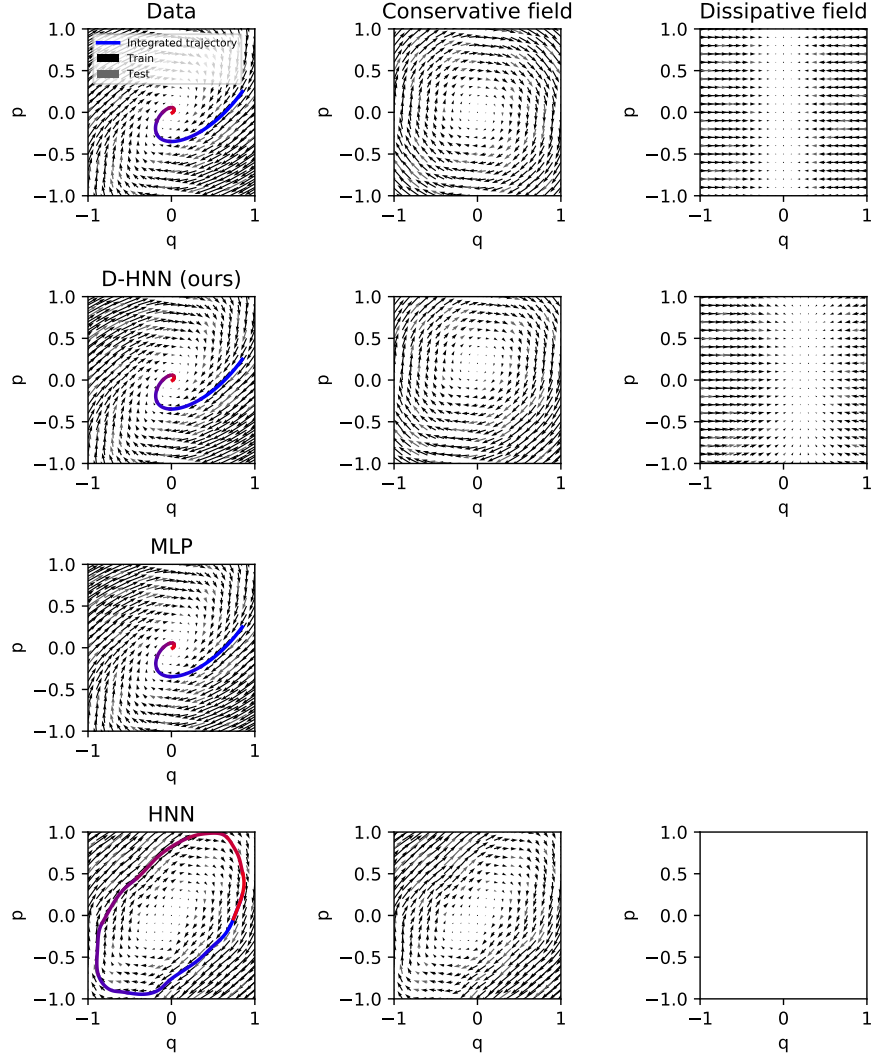
Figure 6: Training a Dissipative Hamiltonian Neural Network (D-HNN) and several baseline models on a damped spring task. **Row 1.** We decompose the original dataset by interpolating the training data onto a grid with a nearest-neighbors approach and then decomposing the field into two components. In practice, we accomplish this using a few hundred iterations of the Gauss-Seidel method to solve Poisson's equation. **Row 2.** We train a D-HNN on the same data. The first half of this model looks exactly like an HNN. The second half also looks like an HNN, except we use the gradient of the scalar field directly rather than rearranging it according to Hamilton's equations. This gives us a trainable *dissipative* field which is able to model the dissipative components of the damped spring's dynamics. Summing these two fields, we obtain an accurate model of the dynamics of the system. **Row 3.** We train the baseline model (MLP) on the training set; this model gives a good fit and can be integrated as a Neural ODE, but it cannot be used to decompose the field into conservative and dissipative components. **Row 4.** We train a Hamiltonian Neural Network (HNN) on the same dataset and find that it is only able to model the conservative component of the system's dynamics. In other words, it strictly enforces conservation of energy in a scenario where energy is not actually conserved.