Port-Hamiltonian Neural Networks for Learning Coupled Systems and Their Interactions

Razmik Arman Khosrovian
Graduate School of Engineering Science
Osaka University
Toyonaka, Osaka
khosrovian@hopf.sys.es.osaka-u.ac.jp

Takaharu Yaguchi
Graduate School of Science
Kobe University
Kobe, Hyogo
yaguchi@pearl.kobe-u.ac.jp

Takashi Matsubara
Faculty of Information Science and Technology
Hokkaido University
Sapporo, Hokkaido
matsubara@ist.hokudai.ac.jp

Abstract

Recent advances in deep learning have shown its effectiveness in modeling physical phenomena, even when the governing equations are unknown. However, current studies mainly focus on mechanical systems, often overlooking other physical domains, such as electrical systems. Furthermore, existing methods tend to treat systems with multiple elements as a single entity, making it difficult to capture the complex interactions in coupled systems. To address these challenges, we propose a neural network framework based on the port-Hamiltonian formulation, which incorporates modularity and interactions between elements in the systems being modeled and is applicable to electrical circuits. Our experimental results demonstrate that our method is capable of handling various experimental scenarios that existing methods cannot address, while also improving modeling and prediction accuracy. Moreover, by analyzing the modeling results, we can identify the interactions between elements, providing interpretable results that facilitate understanding and further investigations.

1 Introduction

In recent years, deep learning has gained attention as an effective approach for modeling unknown physical phenomena, particularly when the governing equations are not explicitly available [2]. Especially, methods like Hamiltonian neural networks (HNNs) incorporate physical laws such as the conservation of energy as prior knowledge and have led to significant improvements in the performance of data-driven modeling [11]. Numerous approaches that adhere to fundamental physical principles have been proposed, including Lagrangian neural networks (LNNs) and Dissipative SymODEN [4, 19]. However, current studies mainly focus on mechanical systems, often neglecting other physical principles like Kirchhoff's laws in electrical circuits. Furthermore, many models treat systems with multiple elements as a single entity, making it challenging to capture complex interactions in coupled systems like robot arms or electrical circuits composed of multiple circuit elements [16, 18].

To address these challenges, this paper proposes port-Hamiltonian neural networks (port-HNNs), a physics model based on the port-Hamiltonian framework [3, 14, 16]. This model unifies the description of various physical phenomena across domains as well as identifying the coupling

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

Table 1: Comparison between Related Methods.

	Energy Dissipation	External Inputs	Constraints	Identifying Coupling Patterns
Neural ODE [2]	(implicitly)		(implicitly)	
HNN [11]				
Dissipative SymODEN [19]	✓	✓		
Constrained HNN [10]			(only holonomic)	
port-HNN (proposed)	✓	✓	✓	✓

and constraints between elements in the systems being modeled. Port-HNNs improve predictive performance, identify interactions between elements of the system being modeled, and provide more interpretable models.

2 Port-Hamiltonian Neural Networks

Background and Related Work There is a growing need to model dynamical systems describable by ordinary differential equations (ODEs) from data to aid in prediction and control. In this context, neural ordinary differential equations (neural ODEs) have been proposed to approximate the right-hand side f of an ODE $\dot{\boldsymbol{u}} = f(\boldsymbol{u})$ using neural networks, and to predict future states through numerical integration. Note that $\boldsymbol{u}$ is the state of the dynamics and $\dot{}$ denotes time derivative.

If it is known that the dynamical system being modeled adheres to physical laws, such as the conservation of energy, leveraging this prior knowledge can lead to more accurate and effective modeling. The Hamiltonian formulation typically describes an energy-conservative physical system using the generalized coordinates $\boldsymbol{q}$ and the generalized momenta $\boldsymbol{p}$ as its state. This formulation provides the equations of motion called Hamilton's equation, written as $\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} O & I \\ -I & O \end{bmatrix} \nabla H(\boldsymbol{q}, \boldsymbol{p})$, where ∇ denotes the gradient operator, and H is a function that represents the system energy called the Hamiltonian H. It is easily confirmed that this equation preserves the Hamiltonian H because $\dot{H} = \nabla H^\top \begin{bmatrix} O & I \\ -I & O \end{bmatrix} \nabla H = 0$. The original HNNs have leverage this form, where a neural network is trained to approximate the Hamiltonian H. See also Table 1 for comparison.

The aforementioned formulation, however, does not handle systems with energy dissipation due to friction or external inputs applied. To describe such systems, many studies have employed the following form [5, 6, 15, 19]: $\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} O & I \\ -I & -D(\boldsymbol{q}) \end{bmatrix} \nabla H(\boldsymbol{q}, \boldsymbol{p}) + \begin{bmatrix} O \\ G(\boldsymbol{q}) \end{bmatrix} \boldsymbol{F}$, where the submatrix $D(\boldsymbol{q})$ represents the dissipation, $\boldsymbol{F}$ denotes the external input, and $G(\boldsymbol{q})$ denotes the gain of $\boldsymbol{F}$. $D(\boldsymbol{q})$ and $G(\boldsymbol{q})$ are approximated by neural networks similarly to $H(\boldsymbol{q}, \boldsymbol{p})$.

However, this formulation also presents several challenges. First, the coordinates q_i and momenta p_i are paired, which means that it requires the absence of redundancy in measurements. When multiple springs are attached to a single mass point, the states of all the springs are consolidated into a single position, and the unique characteristics of the springs are abstracted into a single potential energy function. This makes the learning process more difficult and complicates the interpretation and application of the results. In some studies, the matrix that precedes $\nabla H(\boldsymbol{q}, \boldsymbol{p})$ is expressed in non-canonical form [1, 9], enabling the modeling of systems where q_i and p_i are not strictly paired. Even so, these methods require the structure of interactions between system components to be explicitly known, and this may limit their applicability. That being said, this formulation shares the remaining challenges of the original formulation.

Similar to the first challenge, the submatrices $D(\boldsymbol{q})$ and $G(\boldsymbol{q})$, which represent energy dissipation and input gain, respectively, are abstracted as functions, making it difficult to interpret how dampers or external forces interact with other elements. Moreover, this formulation is unable to accommodate externally imposed velocities, such as those in models of buildings subjected to ground motion, or externally imposed currents, introduced by current sources in electrical circuits. These limitations significantly restrict its application to systems such as electric circuits. While constrained HNNs can handle the systems with constraints [10], they are limited to the constraints raised from conserved quantities in the coordinates $\boldsymbol{q}$.

Table 2: Correspondence between Elements

Domain	**Mechanical**		**Electrical**	
Elements	**Potential**	**Kinetic**	**Electric**	**Magnetic**
state (integral of flow)	displacement q	momentum p	electric charge Q	magnetic flux ϕ
flow (input)	velocity v	force f	current I	voltage V
effort (output)	force f	velocity v	voltage V	current I
energy-conservative	spring k	mass m	capacitor C	inductor L
energy-dissipative	damper d	–	resistor R	resistor G
external input	external force e	moving boundary b	voltage source E	current source J

Port-Hamiltonian Neural Networks To overcome these difficulties, inspired by the generalized bond graph formulation [8, 16], we propose the port-Hamiltonian neural networks (port-HNNs) in the following form:

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \\ \hline \boldsymbol{f}_{R1} \\ \boldsymbol{f}_{R2} \\ \hline \boldsymbol{f}_{I1} \\ \boldsymbol{f}_{I2} \end{bmatrix} = \left[\begin{array}{cc|cc|cc} 0 & B & 0 & R_v & 0 & E_v \\ -B^\top & 0 & R_f & 0 & E_f & 0 \\ \hline 0 & -R_f^\top & 0 & 0 & 0 & I_v \\ -R_v^\top & 0 & 0 & 0 & I_f & 0 \\ \hline 0 & -E_f^\top & 0 & -I_f^\top & 0 & 0 \\ -E_v^\top & 0 & -I_v^\top & 0 & 0 & 0 \end{array}\right] \begin{bmatrix} \nabla_{\boldsymbol{q}} H(\boldsymbol{q}, \boldsymbol{p}) \\ \nabla_{\boldsymbol{p}} H(\boldsymbol{q}, \boldsymbol{p}) \\ \hline \boldsymbol{e}_{R1} \\ \boldsymbol{e}_{R2} \\ \hline \boldsymbol{e}_{I1} \\ \boldsymbol{e}_{I2} \end{bmatrix}. \tag{1}$$

H is the Hamiltonian, and $(\boldsymbol{q}, \boldsymbol{p})$ denote to local coordinates and momenta, respectively, as before.

The variables $(\dot{\boldsymbol{q}}, \dot{\boldsymbol{p}}, \boldsymbol{f}_{R1}, \boldsymbol{f}_{R2}, \boldsymbol{f}_{I1}, \boldsymbol{f}_{I2})$ on the left-hand side are referred to as flows, whereas the variables $(\nabla_{\boldsymbol{q}} H(\boldsymbol{q}, \boldsymbol{p}), \nabla_{\boldsymbol{p}} H(\boldsymbol{q}, \boldsymbol{p}), \boldsymbol{e}_{R1}, \boldsymbol{e}_{R2}, \boldsymbol{e}_{I1}, \boldsymbol{e}_{I2})$ on the right-hand side are referred to as efforts. In this formulation, the coordinates $\boldsymbol{q}$ do not represent the positions of masses but rather the displacements of the springs, and the number of springs (the size of $\boldsymbol{q}$) may differ from the number of masses (the size of $\boldsymbol{p}$). Intuitively speaking, the flows are the inputs to elements in the target system, and the efforts are the outputs from elements.

Consider, for example, a spring. The input to the spring is the rate of change of its displacement $\boldsymbol{q}$, represented by $\dot{\boldsymbol{q}}$. The output from the spring is the force generated by the spring as it changes, which is expressed as $\nabla_{\boldsymbol{q}} H(\boldsymbol{q}, \boldsymbol{p})$. A spring is considered as an element that converts velocity to force. A mass is, on the other hand, an element that converts force to velocity. The masses and springs are connected and exchange velocity and force with each other, and this coupling pattern is represented by the submatrix B; each non-zero element suggests the coupling between the corresponding elements. In electrical circuits, each coordinate q_i represents to the electric charge of each circuit element, with capacitors analogous to springs and inductors to masses. The correspondences are summarized in Table 2.

Elements denoted by the subscripts $_{R1}$ and $_{R2}$ are elements that dissipate energy, such as dampers and resistors. Elements $_{R1}$ convert velocity to force as $\boldsymbol{e}_{R1} = R_1(\boldsymbol{f}_{R1})$, whereas elements $_{R2}$ convert force to velocity $\boldsymbol{e}_{R2} = R_2(\boldsymbol{f}_{R2})$. If the element is a linear damper, the conversion is $e = -df$. Hence, the outputs from elements $_{R1}$ may be inputted to the masses, but not to the springs, which is represented by the submatrix R_f. Also, the masses may define the velocities of elements $_{R1}$, which is represented by the submatrix $-R_f^\top$. Elements denoted by the subscripts $_{I1}$ and $_{I2}$ represent the external inputs, where elements $_{I1}$ generate forces (that is, they are external forces or voltage sources) and elements $_{I2}$ give velocities (for example, they are moving boundaries or current sources). Each submatrix represents the interactions between the corresponding elements. The elements of the submatrices may take values 1, 0, or -1, similar to an adjacency matrix, but are not limited to these. Forces may be distributed across multiple axes or even across multiple objects.

Therefore, by learning not only the Hamiltonian H but also the characteristics of each energy-dissipating elements, R_1 and R_2, along with the submatrices from data, we can identify the interactions underlying the target system as well as reproducing the system's dynamics.

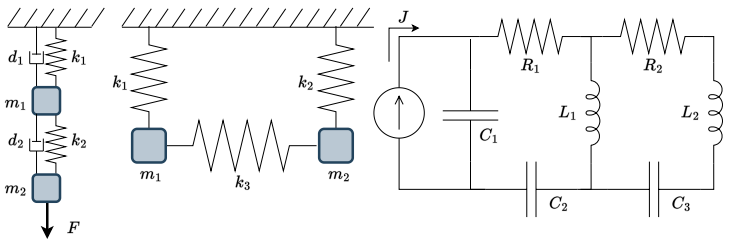

Figure 1: Systems for Task 1 (left), Task 2 (middle), and Task 3 (right).

Table 3: Experimental Results

	task 1		task 2		task 3	
Model	VPT	MSE	VPT	MSE	VPT	MSE
Neural ODE	0.11	51.84	0.43	1040.22	0.38	4.05
Dissipative SymODEN	0.08	15.82	–	–		
Port-HNN	**0.24**	**3.98**	**0.69**	**9.18**	**0.56**	**1.83**
	$\theta = 10^{-5}$	$(\times 10^{-7})$	$\theta = 10$	$(\times 10^{-1})$	$\theta = 1.0$	$(\times 10^{-1})$

3 Experiments and Results

Experimental Setting In this section, we evaluated the performance of our proposed port-HNN through several benchmark tasks, summarized in Figure 1: namely, (task 1) a one-dimensional system consisting of two consecutive mass-spring-damper systems with an applied external force, (task 2) a two-dimensional system where three springs and two masses are coupled, and (task 3) an electrical circuit consisting of three capacitors, two inductors, two resistors, and one alternating-current source. The state is composed of the displacements of springs, the velocities of masses, the electric chargers of capacitors, and the currents through the inductors. See Appendix A for details. We generated the ground-truth data using the Dormand-Prince method with the time step size $\Delta t = 0.01$ [7]. The training dataset consisted of randomly initialized 1,000 trajectories, each spanning 500 steps, while the evaluation dataset consisted of randomly initialized 10 trajectories, each spanning 10,000 steps.

We implemented the port-HNN, the neural ODE, and the dissipative SymODEN for comparison. We trained these models by minimizing the L_2 loss of the time derivatives $(\dot{\boldsymbol{q}}, \dot{\boldsymbol{p}})$ of states $(\boldsymbol{q}, \boldsymbol{p})$. For evaluating models, we integrated these models from the initial conditions using the Dormand-Prince method, and calculated the mean squared errors (MSEs) between the ground truth trajectories and the predicted states. In addition, we used the valid prediction time (VPT) [17], which is defined as the ratio of the time until the MSE exceeds a certain threshold θ to the total length of the trajectory. A smaller MSE and a larger VPT indicate better performance.

Results We summarized the results in Table 3. Our proposed port-HNN outperformed the comparison methods in all three tasks. Figure 2 depicts an example of the predicted states for task 3. It demonstrates that the port-HNN predicted the future states of the target systems with smaller errors for a long time. We visualized an example result of the submatrices learned in task 1:

$$B = \begin{bmatrix} 0.01785 & 0.00000 \\ -0.01785 & 0.01786 \end{bmatrix}, R_f = \begin{bmatrix} 0.27819 & 0.13647 \\ 0.00046 & -0.13618 \end{bmatrix}, E_f = \begin{bmatrix} 0.00025 \\ 0.99976 \end{bmatrix}. \tag{2}$$

The upper row of the submatrix B suggests that the spring k_1 is coupled with the mass m_1 but not with m_2. The bottom row suggests that the spring k_2 is coupled with both the masses, m_1 and m_2, in opposite directions. Also, the submatrix R_f suggests that the dumpers d_1 and d_2 are coupled with the masses in the same patterns as the springs. Finally, the submatrix E_f suggests that the external input is coupled only with the mass m_2. Please note that the scale of the matrix elements cancel out with the characteristics of the corresponding elements. We also visualized an example result of the submatrix B learned in task 2:

$$B = \left[\begin{array}{cc|cc} 0.9999 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.9411 & 0.0000 & 0.0000 \\ \hline 0.0000 & 0.0000 & 1.0588 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0588 \\ \hline -0.9999 & 0.0000 & 0.9999 & 0.0000 \\ 0.0000 & -0.9411 & 0.0000 & 1.0588 \end{array}\right] = \begin{bmatrix} (1,1) & (1,2) \\ (2,1) & (2,2) \\ (3,1) & (3,2) \end{bmatrix}. \tag{3}$$

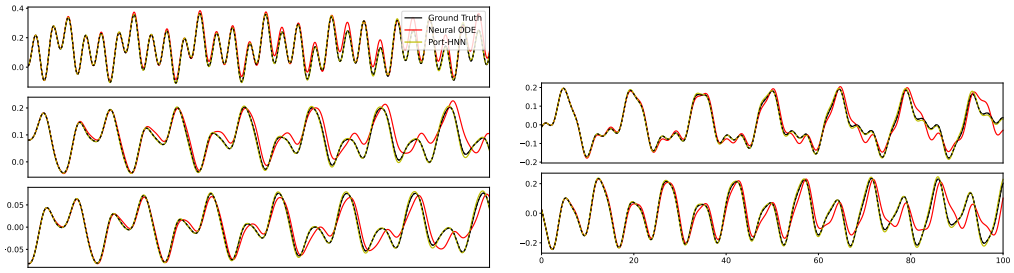

Figure 2: The ground truth and example predicted results of the states in task 3. (left) The capacitors C_1, C_2, C_3 from top to bottom. (right) The inductors L_1, L_2 from top to bottom.

Due to the 2-dimensional state space, the 2×2 submatrix (i, j) represents the coupling between the i-th spring k_i and j-th mass m_j. We can see that the spring k_1 is connected to the mass m_1, the spring k_2 is connected to the mass m_2, and the spring k_3 is connected to both the masses, m_1 and m_2. These results are consistent with the diagrams in Figure 1. This demonstrates that the port-HNN is not only effective at modeling and predicting system dynamics, but also provides interpretable insights into the internal structure of the dynamical systems.

References

[1] Thomas Beckers, Jacob Seidman, Paris Perdikaris, and George J. Pappas. Gaussian process port-hamiltonian systems: Bayesian learning with physics prior. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 1447–1453, 2022.

[2] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1–19, 2018.

[3] Karim Cherifi. An overview on recent machine learning techniques for port hamiltonian systems. *Physica D: Nonlinear Phenomena*, 411:132620, 2020.

[4] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian Neural Networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations (DeepDiffEq)*, pages 1–9, March 2020.

[5] 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), September 2021.

[6] Luca Di Persio, Matthias Ehrhardt, and Sofia Rizzotto. Integrating Port-Hamiltonian Systems with Neural Networks: From Deterministic to Stochastic Frameworks. *arXiv*, March 2024.

[7] J.R. Dormand and P.J. Prince. A reconsideration of some embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 15(2):203–211, 1986.

[8] Vincent Duindam, Alessandro Macchelli, Stefano Stramigioli, and Herman Bruyninckx. *Modeling and Control of Complex Physical Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[9] Sølve Eidnes, Alexander J. Stasik, Camilla Sterud, Eivind Bøhn, and Signe Riemer-Sørensen. Pseudo-Hamiltonian Neural Networks with State-Dependent External Forces. *Physica D: Nonlinear Phenomena*, 446:133673, April 2023.

[10] Marc Finzi, Ke Alexander Wang, and Andrew Gordon Wilson. Simplifying Hamiltonian and Lagrangian Neural Networks via Explicit Constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[11] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1–16, 2019.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, pages 1–15, December 2015.

[13] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, pages 1–16, 2017.

[14] Subramanya P. Nageshrao, Gabriel A. D. Lopes, Dimitri Jeltsema, and Robert Babuška. Port-hamiltonian systems in adaptive and learning control: A survey. *IEEE Transactions on Automatic Control*, 61(5):1223–1238, 2016.

[15] Cyrus Neary and Ufuk Topcu. Compositional Learning of Dynamical System Models Using Port-Hamiltonian Neural Networks. In *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, pages 679–691. PMLR, June 2023.

[16] Arjan van der Schaft and Dimitri Jeltsema. Port-Hamiltonian Systems Theory: An Introductory Overview. *Foundations and Trends® in Systems and Control*, 1(2):173–378, 2014.

[17] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos. Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126:191–217, 2020.

[18] Hiroaki Yoshimura and Jerrold E. Marsden. Dirac structures in Lagrangian mechanics Part I: Implicit Lagrangian systems. *Journal of Geometry and Physics*, 57(1):133–156, 2006.

[19] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Dissipative SymODEN: Encoding Hamiltonian Dynamics with Dissipation and Control into Deep Learning. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations (DeepDiffEq)*, pages 1–6, 2020.

A Details of Experiments

In task 1, each spring was set to be nonlinear, generating the force $-kq - bq^3$, where k, b are parameters. Each damper was also nonlinear, generating a resistive force $-dv^{1/3}$ for the damper velocity v. We set $m_1 = 5.0, m_2 = 3.0$ for masses, $k_1 = 0.5, k_2 = 3.0, b_1 = 9.0, b_2 = -0.5$ for springs, and $d_1 = 0.11, d_2 = 0.08$ for dampers. The initial displacements and velocities were sampled from uniform distributions $U(-0.5, 0.5)$ and $(-0.1, 0.1)$, respectively. Additionally, an external force was applied to one of the masses. It was written as a sine wave $A\sin(wt + b)$, where the amplitude A, the angular velocity w, and the phase b were randomly sampled from the uniform distributions $U(0.05, 0.3)$, $U(-10, 10)$, and $U(-\pi, \pi)$, respectively.

In task 2, each spring was nonlinear in the same way as in task 2, but was allowed to move in the 2-dimensional space. Three springs have a nominal state space of $3 \times 2 = 6$ dimensions, but because they are connected to two masses, the effective degrees of freedom were reduced to $2 \times 2 = 4$ dimensions. This reduction was assumed to be unknown. Overall, the system had a 10-dimensional state space, which includes the $2 \times 2 = 4$-dimensional velocity of the masses. We set $m_1 = 5.0, m_2 = 3.0$ for masses and $k_1 = 2.5, k_2 = 3.0, k_3 = 2.1, b_1 = 7.4, b_2 = -0.5, b_3 = 5.1$ for springs. The initial displacements and velocities were sampled from uniform distributions $U(-0.5, 0.5)$ and $(-0.1, 0.1)$, respectively.

In task 3, each resistor was nonlinear, with the voltage increasing in proportion to the cube of the current with the coefficient r. We used $C_1 = 3.0, C_2 = 2.0, C_3 = 0.5$ for capacitors, $L_1 = 2.9, L_2 = 4.5$ for inductors, and $r_1 = 1.3, r_2 = 2.1$ for resistors. The initial electric charges of capacitors and currents through inductors were sampled from uniform distributions $U(-0.5, 0.5)$ and $(-0.1, 0.1)$, respectively. The current source was defined as $A\sin(wt + b)$, where the amplitude A, the angular velocity w, and the phase b were randomly sampled from the uniform distributions $U(0.05, 0.3)$, $U(-5, 5)$, and $U(-\pi, \pi)$, respectively.

We used a 3-layer fully-connected neural network with 200 hidden units and the hyperbolic tangent activation function for modeling all functions, namely the right-hand side of the neural ODE, the Hamiltonian H, and the characteristics of the energy-dissipative elements $_{R1}$ and $_{R2}$. In our formulation, we can consider each energy-conservative element having its own Hamiltonian H. We assume that the capacitors and inductors behave linearly and that the kinetic energy of the masses is proportional to the square of their velocity. Therefore, the Hamiltonians H corresponding to these elements do not need to be modeled using neural networks; instead, they can be modeled as quadratic functions with learnable parameters, such as capacitances, inductances, and masses. By utilizing these relationships, we can interconvert momentum and velocity of mass, voltage and electric charge of capacitor, as well as current and magnetic flux of inductor.

Each model was trained using the Adam optimizer [12] for 40,000 iterations for tasks 1 and 2, and for 10,000 iterations for task 3. We used the parameters $(\beta_1, \beta_2) = (0.9, 0.999)$ and a batch size of 200. The learning rate was initialized to 10^{-3} and decayed to zero with cosine annealing [13].

The proposed port-HNN is defined with the displacements of the springs as the coordinates $\boldsymbol{q}$, while the dissipative SynODEN is with the absolute positions of the masses as the coordinates $\boldsymbol{q}$. We used the absolute positions of the masses for the dissipative SymODEN and neural ODE in task 1, while we used the displacements of the springs for other combination of tasks and methods.