Proceedings of Machine Learning Research 1–16

Compositional Learning of Dynamical System Models Using Port-Hamiltonian Neural Networks

Cyrus Neary CNEARY@UTEXAS.EDU
Ufuk Topcu UTOPCU@UTEXAS.EDU
The University of Texas at Austin, United States

Abstract

Many dynamical systems—from robots interacting with their surroundings to large-scale multiphysics systems—involve a number of interacting subsystems. Toward the objective of learning composite models of such systems from data, we present i) a framework for compositional neural networks, ii) algorithms to train these models, iii) a method to compose the learned models, iv) theoretical results that bound the error of the resulting composite models, and v) a method to learn the composition itself, when it is not known a priori. The end result is a modular approach to learning: neural network submodels are trained on trajectory data generated by relatively simple subsystems, and the dynamics of more complex composite systems are then predicted without requiring additional data generated by the composite systems themselves. We achieve this compositionality by representing the system of interest, as well as each of its subsystems, as a *port-Hamiltonian neural network* (PHNN)—a class of neural ordinary differential equations that uses the port-Hamiltonian systems formulation as inductive bias. We compose collections of PHNNs by using the system's physics-informed *interconnection structure*, which may be known a priori, or may itself be learned from data. We demonstrate the novel capabilities of the proposed framework through numerical examples involving interacting spring-mass-damper systems. Models of these systems, which include nonlinear energy dissipation and control inputs, are learned independently. Accurate compositions are learned using an amount of training data that is negligible in comparison with that required to train a new model from scratch. Finally, we observe that the composite PHNNs enjoy properties of port-Hamiltonian systems, such as cyclo-passivity—a property that is useful for control purposes.

Keywords: Physics-informed machine learning, port-Hamiltonian neural networks, neural ordinary differential equation, compositional deep learning

1. Introduction

Deep learning methods that use physics-based knowledge as inductive bias have recently shown promise in learning dynamical system models that respect physical laws and that generalize beyond the training dataset (Djeumou et al., 2022a; Menda et al., 2019; Gupta et al., 2020; Cranmer et al., 2020; Greydanus et al., 2019; Finzi et al., 2020; Zhong et al., 2021a). These methods, which often use neural networks to parametrize select terms in differential operators, are able to learn complex relationships from data while also yielding models that are compact and interpretable.

However, there remain barriers to the deployment of such algorithms in engineering applications. Many systems, from robots interacting with their surroundings to large-scale multiphysics systems, involve large numbers of interacting components. These interactions between subsystems

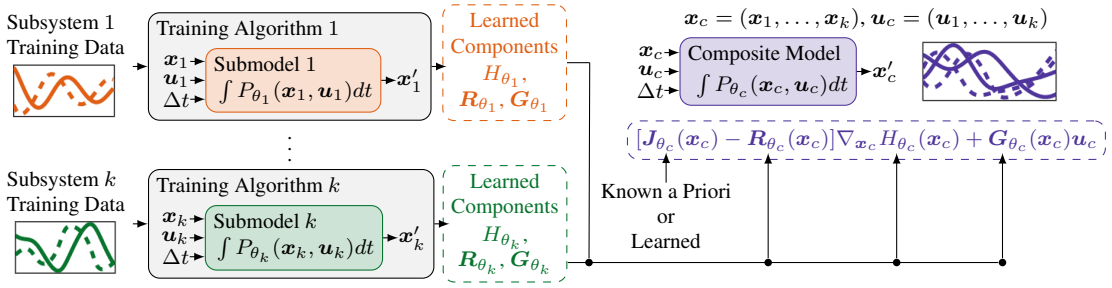

Figure 1: An illustration of the compositional learning framework. We train separate port-Hamiltonian neural networks (PHNNs) on data generated by individual subsystems, presented in §4. We then compose these submodels to construct another PHNN that models the composite system, presented in §5.

can increase the complexity of the overall system's dynamics, rendering monolithic approaches to learning—which capture the entire system using a single model learned from data—challenging.

We present a framework and algorithms for learning and composing neural network models of dynamical systems. The framework models individual subsystems independently, and uses physics-informed interfaces between these submodels to capture their interactions. This compositional approach to learning provides a number benefits and novel capabilities that would not otherwise be possible. Firstly, it simplifies the learning problems to be solved. Submodels are trained on trajectory data generated by relatively simple subsystems. The dynamics of more complex composite systems are then predicted without requiring additional training. Secondly, it provides a modular framework for data-driven modeling. Previously learned component models can be composed in new ways to simulate different composite systems. Finally, it provides a natural way to compose data-driven models with models derived from first principles.

We achieve this compositionality by representing the system of interest, as well as each of its subsystems, as a *port-Hamiltonian neural network* (PHNN)—a class of deep learning models that use the port-Hamiltonian systems formalism (Duindam et al., 2009; Van Der Schaft et al., 2014) to inform the model's structure. More specifically, PHNNs parametrize each subsystem's Hamiltonian function, as well as how it dissipates energy, interacts with other subsystems, and how it responds to control inputs. We enforce known properties of the dissipation and interaction terms through the model's construction. The PHNN's output is obtained by numerically integrating a differential equation involving all of these terms to predict the system's state at a future time.

Using the physics-informed structure provided by the PHNN, we present a method to compose collections of PHNNs in order to obtain models of the corresponding composite systems, and we provide upper bounds on this composite model's prediction errors. Figure 1 illustrates the approach. The composite system's Hamiltonian, dissipation term, and control input term are all obtained by combining the corresponding terms from the learned submodels. Interactions between the subsystems are captured by the *interconnection structure* of the composite system, which may be known a priori, or may itself be learned from data. In many cases this interconnection structure is given by a constant linear operator; we accordingly present a method to learn it via linear regression. In the general case, it may instead be parameterized using a neural network.

We demonstrate the novel capabilities of the proposed framework through numerical examples involving interacting spring-mass-damper systems. Models of these systems, which include nonlin-

ear energy dissipation and control inputs, are learned independently. The dynamics of the composite system are accurately predicted without additional training. If the system's interconnection structure is unknown, we demonstrate that an accurate composition may be learned using an amount of training data that is negligible in comparison with that required to train a new model from scratch. Finally, we empirically observe that the proposed compositions of PHNNs exhibit the property of passivity—a property of port-Hamiltonian systems that is useful for control purposes.

2. Related Work

The port-Hamiltonian formulation of dynamical systems provides a rich mathematical framework that enables compositional modeling (Duindam et al., 2009; Van Der Schaft et al., 2014). This framework can be applied quite generally and has been used to model fluid-structure interactions (Cardoso-Ribeiro et al., 2015), aerial vehicles in contact scenarios (Rashad et al., 2021), and coupled gas and electricity distribution networks (Strehle et al., 2018). Furthermore, a wealth of existing methods and analysis for the nonlinear control of port-Hamiltonian systems already exists (Van Der Schaft, 2020). However, deriving a precise system model in port-Hamiltonian form can be challenging in many practical applications (Nageshrao et al., 2015; Cherifi et al., 2019). Furthermore, methods for data-driven identification and control of port-Hamiltonian systems have not yet been extensively explored (Nageshrao et al., 2015; Cherifi et al., 2022).

The inclusion of physics-based knowledge into neural network models of dynamical systems has, however, been studied extensively over the past several years. In particular, neural ordinary differential equations (NODEs) (Chen et al., 2018) provide a natural approach to incorporate physics-based knowledge as inductive bias in deep learning (Zhong et al., 2021a; Rackauckas et al., 2020). By using neural networks to parametrize differential equations, as opposed to directly fitting the available trajectory data, NODEs allow the user to harness an existing wealth of knowledge from applied mathematics, physics, and engineering (Djeumou et al., 2022a; Cranmer et al., 2020; Lutter et al., 2019; Gupta et al., 2020; Roehrl et al., 2020; Zhong et al., 2021b; Shi et al., 2019).

Of particular relevance to our work, *Hamiltonian neural networks* use the Hamiltonian formulation of dynamics to inform the structure of a neural ODE (Greydanus et al., 2019; Matsubara et al., 2020; Toth et al., 2020; Finzi et al., 2020). However, Hamiltonian-based models necessarily represent closed systems. By contrast, we study systems involving energy exchange, energy dissipation, and control inputs. Meanwhile, Xu et al. (2021); Furieri et al. (2022); Plaza et al. (2022) use neural networks to parametrize controllers for port-Hamitlonian systems. However, these works do not learn dynamics models—they assume the dynamics to be known and focus on control.

More closely related to our work, several recent papers also study neural ODEs that have a port-Hamiltonian structure (Zhong et al., 2020; Desai et al., 2021; Eidnes et al., 2022; Duong and Atanasov, 2021). However, none of these works study how physics-based knowledge may be used to compose deep learning models. By contrast, the primary focus of this work is to develop a framework, theoretical results, and methods that enable such compositional learning algorithms.

3. Port-Hamiltonian Systems

Port-Hamiltonian (PH) systems provide a versatile framework that enables the modeling and analysis of complex networks of interacting subsystems. Conceptually, PH systems are represented by their Hamiltonian functions, by their energy dissipation terms, and by a mathematical description

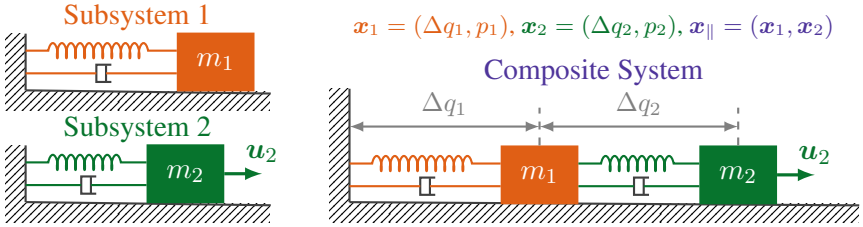

Figure 2: An illustrative coupled spring-mass-damper example. Two subsystems (left), with different spring and damping constants, are connected to obtain the composite system (right).

of the power-conserving interactions of their subsystems, called a *Dirac structure*. We refer to Van Der Schaft et al. (2014); Duindam et al. (2009) for further details.

In this work, we consider lumped parameter PH systems expressed in *explicit state-input-output* form; the system's state may be represented as a finite-dimensional vector, and its dynamics are given by equation 1. PH systems may be expressed in this form whenever there are no algebraic constraints on the system's state variables (Donaire and Junco, 2009; Dai et al., 2019).

$$\dot{\boldsymbol{x}} = [\boldsymbol{J}(\boldsymbol{x}) - \boldsymbol{R}(\boldsymbol{x})] \nabla_{\boldsymbol{x}} H(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u}, \quad \boldsymbol{y} = \boldsymbol{G}(\boldsymbol{x})^T \nabla_{\boldsymbol{x}} H(\boldsymbol{x}) \tag{1}$$

Here, $\boldsymbol{x} \in \mathbb{R}^n$ is the n-dimensitonal vector representing the system's state, $H(\boldsymbol{x})$ is the system's Hamiltonian function, $\boldsymbol{J}(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ is the skew symmetric interconnection matrix (i.e. $\boldsymbol{J}(\boldsymbol{x}) = -\boldsymbol{J}(\boldsymbol{x})^T$ for every $\boldsymbol{x} \in \mathbb{R}^n$), $\boldsymbol{R}(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ is the symmetric positive semi-definite dissipation matrix (i.e. $\boldsymbol{R}(\boldsymbol{x}) = \boldsymbol{R}(\boldsymbol{x})^T$ and $\tilde{\boldsymbol{x}}^T \boldsymbol{R}(\boldsymbol{x}) \tilde{\boldsymbol{x}} \geq 0$ for every $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathbb{R}^n$), $\boldsymbol{G}(\boldsymbol{x}) \in \mathbb{R}^{n \times m}$ is the control input matrix, $\boldsymbol{u} \in \mathbb{R}^m$ is the m-dimensional control input vector, and $\boldsymbol{y} \in \mathbb{R}^m$ is the corresponding output vector. Intuitively, while $H(\boldsymbol{x})$ describes the system's energy in terms of its state $\boldsymbol{x}$, $\boldsymbol{J}(\boldsymbol{x})$ encodes the energy-conserving interactions between the various components of the system and $\boldsymbol{R}(\boldsymbol{x})$ encodes how these components dissipate energy.

3.1. An Illustrative Running Example: The Coupled Mass-Spring-Damper

Let PHS_1 and PHS_2 represent the two subsystems on the left of Figure 2, and let PHS_c represent the system resulting from their composition, illustrated on the right. Similarly, let $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{u}_1$, and $\boldsymbol{u}_2$ denote the subsystem states and control inputs, and let $\boldsymbol{x}_c := (\boldsymbol{x}_1, \boldsymbol{x}_2)$, $\boldsymbol{u}_c := (\boldsymbol{u}_1, \boldsymbol{u}_2)$ denote the state and control input of the composite system.

We assume that the damping forces are nonlinear $F_{damp} = b\dot{\boldsymbol{q}}^3$, similar to the example presented by Lopes et al. (2015). We define the subsystem states as $\boldsymbol{x}_i = (\Delta \boldsymbol{q}_i, \boldsymbol{p}_i)$ for $i = 1, 2$, where $\Delta \boldsymbol{q}_i$ is the elongation of the spring and $\boldsymbol{p}_i$ is the momentum of the mass. The dynamics of the subsystems may then be represented in the form of equation 1 with $H_i(\boldsymbol{x}_i) = \frac{\boldsymbol{p}_i^2}{2m_i} + \frac{k \Delta \boldsymbol{q}_i^2}{2}$ and with

$$\boldsymbol{J}_i = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad \boldsymbol{R}_i(\boldsymbol{x}_i) = \begin{bmatrix} 0 & 0 \\ 0 & b_i \frac{\boldsymbol{p}_i^2}{m_i^2} \end{bmatrix}, \quad \boldsymbol{G}_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2}$$

The port-Hamiltonian system PHS_c representing the system on the right of Figure 2 may be obtained by *composing* PHS_1 and PHS_2, the port-Hamiltonian reprepresentations of the subsystems. That is, the dynamics of PHS_c may also be written in the form of equation 1, where the composite Hamiltonian is given by $H_c(\boldsymbol{x}_c) = H_1(\boldsymbol{x}_1) + H_2(\boldsymbol{x}_2)$, the dissipation $\boldsymbol{R}_c(\boldsymbol{x}_c)$ and control input

$\boldsymbol{G}_c$ matrices are obtained by stacking the matrices $\boldsymbol{R}_i(\boldsymbol{x}_i)$ and $\boldsymbol{G}_i$ diagonally, and the composite interconnection term $\boldsymbol{J}_c$ is obtained by stacking $\boldsymbol{J}_i$ diagonally and by including an additional pair of off-diagonal interaction terms. These additional entries in $\boldsymbol{J}_c$ encode the coupling between the subsystems; spring 2 exerts a force of $\frac{\partial H_2}{\partial \Delta \boldsymbol{q}_2}$ on mass 1, and mass 1's contribution to the rate of change in the elongation of spring 2 is given by $-\frac{\partial H_1}{\partial \boldsymbol{p}_1}$. The composite system's dynamics are given explicitly in Appendix B.

4. Port-Hamiltonian Neural Networks

Our objective is to train port-Hamiltonian neural networks (PHNN) to predict the dynamics of relatively simple subsystems (e.g., the individual spring-mass-damper systems from the left of Figure 2) and then to compose these learned models in order to simulate more complex systems (e.g., the coupled spring-mass-damper system on the right of Figure 2). In this section we present how to construct and train PHNN submodels. We present a method to compose the learned models in §5.

4.1. Constructing Port-Hamiltonian Neural Networks

A PHNN parametrizes the unknown terms in equation 1, and solves the resulting differential equation in order to predict the system's future states. Let $P(\boldsymbol{x}, \boldsymbol{u})$ denote the right-hand side of the state equation in 1. We use $H_\theta(\boldsymbol{x})$, $\boldsymbol{R}_\theta(\boldsymbol{x})$, and $\boldsymbol{G}_\theta(\boldsymbol{x})$ to denote the parametrizations of the potentially unknown terms in $P(\boldsymbol{x}, \boldsymbol{u})$. We use $P_\Theta(\boldsymbol{x}, \boldsymbol{u})$ to denote the expression that results when each of the unknown terms is replaced with its parametrization, where Θ denotes the collection of all the individual parameter vectors.

Figure 3 illustrates the process of constructing and evaluating a PHNN. The Hamiltonian $H_\theta(\boldsymbol{x})$ is parametrized as a multi-layer perceptron (MLP) and its gradient $\nabla_{\boldsymbol{x}} H_\theta(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ is computed using automatic differentiation. The entries in the input matrix $\boldsymbol{G}_\theta(\boldsymbol{x}) \in \mathbb{R}^{n \times m}$ are either parametrized directly (if $\boldsymbol{G}$ is known to be a constant matrix), or as the output of an MLP (if $\boldsymbol{G}(\boldsymbol{x})$ varies with $\boldsymbol{x}$). The dissipation term $\boldsymbol{R}_\theta(\boldsymbol{x})$ is similarly either parametrized as a constant matrix or as the output of an MLP. However, we additionally enforce the positive semi-definiteness of $\boldsymbol{R}_\theta(\boldsymbol{x})$ by parametrizing its Cholesky decomposition, instead of parametrizing its entries directly. That is, we define $\boldsymbol{R}_\theta(\boldsymbol{x}) := \boldsymbol{L}_\theta(\boldsymbol{x})\boldsymbol{L}_\theta(\boldsymbol{x})^T$ for some parametrized lower-triangular matrix $\boldsymbol{L}_\theta(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ with non-negative diagonal entries.

Note that in this section, similar to all existing works involving port-Hamiltonian neural networks (Greydanus et al., 2019; Zhong et al., 2020; Desai et al., 2021; Eidnes et al., 2022), we assume that the interaction term $\boldsymbol{J}(\boldsymbol{x})$ is known a priori. It is possible to learn a skew-symmetric parametrization of $\boldsymbol{J}(\boldsymbol{x})$ along with all of the other terms in the PHNN. However, doing so necessitates additional considerations that are beyond the scope of the current work, when composing the resulting models.

PHNNs enforce the cyclo-passivity property by construction. The *cylco-passivity* property enjoyed by port-Hamiltonian systems ensures that $dH/dt \leq \boldsymbol{y}^T\boldsymbol{u}$—the rate of change of the system's energy cannot exceed the externally-provided power (Van Der Schaft, 2000). Regardless of the output of the parametrized Hamiltonian $H_\theta(\boldsymbol{x})$, this property is guaranteed by the skew-symmetry of $\boldsymbol{J}(\boldsymbol{x})$ together with the positive semi-definiteness of the dissipation term $\boldsymbol{R}_\theta(\boldsymbol{x})$, which we enforce through the PHNN's construction. When dissipation and control inputs are both present, $dH/dt = h^T\boldsymbol{J}(\boldsymbol{x})h - h^T\boldsymbol{R}_\theta(\boldsymbol{x})h + h^T\boldsymbol{G}_\theta(\boldsymbol{x})\boldsymbol{u} \leq \boldsymbol{y}^T\boldsymbol{u}$. Here, we use h to denote $\nabla_{\boldsymbol{x}} H_\theta(\boldsymbol{x})$.

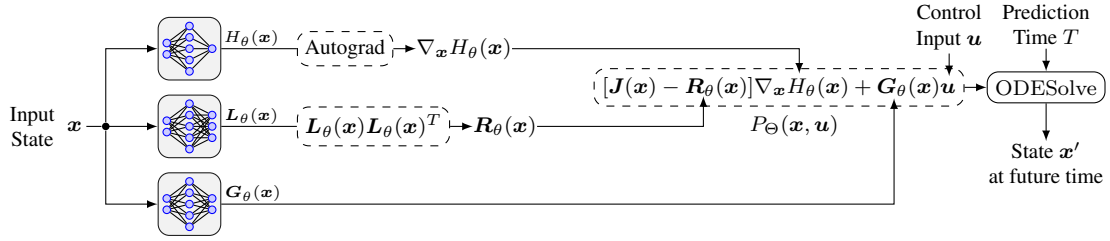

Figure 3: An illustration of the construction and evaluation of a port-Hamiltonian Neural Network.

4.2. Evaluating Port-Hamiltonian Neural Networks

The input to a PHNN is a tuple $(\boldsymbol{x}, \boldsymbol{u}, t, T)$ consisting of the current state $\boldsymbol{x}$, a control input $\boldsymbol{u}$, the current time t, and a prediction time T with $t < T$. The output $\hat{\boldsymbol{x}}_T := PHNN_\Theta(\boldsymbol{x}, \boldsymbol{u}, t, T)$ of the PHNN parametrized by Θ is then given by $\text{ODESolve}(P_\Theta(\cdot, \boldsymbol{u}), \boldsymbol{x}, t, T) \approx \boldsymbol{x} + \int_t^T P_\Theta(\boldsymbol{x}_s, \boldsymbol{u})ds$, where we use the subscript notation $\boldsymbol{x}_t$ to denote $\boldsymbol{x}(t)$. Here, $\text{ODESolve}(P_\Theta(\cdot, \boldsymbol{u}), \boldsymbol{x}, t, T)$ is a numerical solution to the ordinary differential equation specified by $P_\Theta(\boldsymbol{x}, \boldsymbol{u})$ over the window of time $[t, T]$. We note that the particular algorithm used to evaluate $\text{ODESolve}(\cdot)$ influences the model's accuracy and the computational cost of forward evaluations of the model (Djeumou et al., 2022b). In this work, we use fixed-timestep RK4 to evaluate $\text{ODESolve}(\cdot)$ in all experiments.

4.3. Training Port-Hamitlonian Neural Networks

We assume that a finite dataset $\mathcal{D}$ of system trajectories—time-series data of states and control inputs—is available in lieu of the system's model. That is, we are given a set $\mathcal{D} = \{\tau_1, \ldots, \tau_{|\mathcal{D}|}\}$ of trajectories $\tau = \{(\boldsymbol{x}_0, \boldsymbol{u}_0, t_0), \ldots, (\boldsymbol{x}_{|\tau|}, \boldsymbol{u}_{|\tau|}, t_{|\tau|})\}$, where $\boldsymbol{x}_i$ is the state at time t_i, $\boldsymbol{u}_i$ is the control input applied from time t_i until t_{i+1}, and $t_0 < t_1 < \ldots < t_{|\tau|}$ is an increasing sequence of points in time. Note that we are using $P_\Theta(\cdot)$ to parametrize the time derivative $\dot{\boldsymbol{x}}$ of the system's state, which is not explicitly included in the dataset.

Given a dataset $\mathcal{D}$, we construct the loss function $\mathcal{L}(\Theta, \mathcal{D})$ of the PHNN to capture the error in the model's predictions of the future states, as in equation 3, for any given norm.

$$\mathcal{L}(\Theta, \mathcal{D}) = \sum_{\tau_l \in \mathcal{D}} \sum_{(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) \in \tau_l} \|PHNN_\Theta(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i, t_{i+1}) - \boldsymbol{x}_{i+1}\|^2 \tag{3}$$

Finally, we search for local minima of $\mathcal{L}(\Theta, \mathcal{D})$ using gradient-based techniques, where $\nabla_\Theta \mathcal{L}(\Theta, \mathcal{D})$ may be computed using either direct automatic differentiation through $\text{ODESolve}(\cdot)$, or using the adjoint sensitivity method (Pontryagin, 1987; Chen et al., 2018).

5. Composing Port-Hamiltonian Neural Networks

While in §4 we introduced PHNNs and methods to train them, in this section we present a method to compose previously learned PHNNs in order to predict the dynamics of larger composite systems. In the context of the example from §3.1, we use the methods from §4 to learn PHNN models of the individual spring-mass-dampers, and we use the methods in this section to compose these models in order to predict the dynamics of the coupled system.

5.1. Composing Models Using a Known Interconnection Structure

We construct a model of the composite system $PHNN^{\boldsymbol{C}}_{c,\Theta_c}(\cdot)$ by combining the terms of the learned submodels $PHNN_{i,\Theta}(\cdot)$, for $i = 1, \ldots, k$, as illustrated in Figure 1. Let $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ and $\boldsymbol{u}_i \in \mathbb{R}^{m_i}$ represent the state and control input vectors of subsystem i. We define the state and control input vectors of the composite system to be $\boldsymbol{x}_c := \left[\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_k^T\right]^T \in \mathbb{R}^{n_c}$ and $\boldsymbol{u}_c := \left[\boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_k^T\right]^T \in \mathbb{R}^{m_c}$, where $n_c = n_1 + \ldots + n_k$ and $m_c = m_1 + \ldots + m_k$.

The Hamiltonian of the composite system is defined as the sum of the subsystem Hamiltonians, $H_{c,\theta}(\boldsymbol{x}_c) := H_{1,\theta}(\boldsymbol{x}_1) + \ldots + H_{k,\theta}(\boldsymbol{x}_k)$. Note that we assume that each subsystem's Hamiltonian function $H_i(\boldsymbol{x}_i)$ depends only on the corresponding state vector $\boldsymbol{x}_i$. The dissipation and control input terms are defined as $\boldsymbol{R}_{c,\theta}(\boldsymbol{x}_c) := \text{Diag}(\boldsymbol{R}_{1,\theta}(\boldsymbol{x}_1), \ldots, \boldsymbol{R}_{k,\theta}(\boldsymbol{x}_k))$ and $\boldsymbol{G}_{c,\theta}(\boldsymbol{x}_c) := \text{Diag}(\boldsymbol{G}_{1,\theta}(\boldsymbol{x}_1), \ldots, \boldsymbol{G}_{k,\theta}(\boldsymbol{x}_k))$, respectively. Here, we use $\text{Diag}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k)$ to represent the matrix that results from using the submatrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k$ to define the blocks of entries along the matrix diagonal, similarly to as in the example from §3.1. Finally, the interconnection term of the composition is given by $\boldsymbol{J}_c(\boldsymbol{x}_c) := \text{Diag}(\boldsymbol{J}_1(\boldsymbol{x}_1), \ldots, \boldsymbol{J}_k(\boldsymbol{x}_k)) + \boldsymbol{C}(\boldsymbol{x}_c)$, where $\boldsymbol{C}(\boldsymbol{x}_c) \in \mathbb{R}^{n_c \times n_c}$ is a skew symmetric *composition* matrix encoding energy-conserving interactions between the state variables of the various subsystems. Furthermore, we define the blocks of entries along the diagonal of $\boldsymbol{C}(\boldsymbol{x}_c)$ to be zero—the internal subsystem interactions defined by $\boldsymbol{J}_i(\boldsymbol{x}_i)$ should not be altered by $\boldsymbol{C}(\boldsymbol{x}_c)$. We note that the interconnection and dissipation terms of the composite model retain their properties of skew-symmetry and positive semidefiniteness, respectively.

5.2. Learning Compositions of Port-Hamiltonian Neural Network Submodels

If the composition term $\boldsymbol{C}(\boldsymbol{x}_c)$ is unknown, we propose to learn it using additional data gathered from the composite system. That is, in addition to the datasets $\mathcal{D}_1, \ldots, \mathcal{D}_k$ used to train the subsystem models, we assume access to a dataset $\mathcal{D}_c = \{\tau_1^c, \ldots, \tau_{|\mathcal{D}_c|}^c\}$ of trajectories $\tau^c = \{(\boldsymbol{x}_{c,0}, \boldsymbol{u}_{c,0}, t_0), \ldots, (\boldsymbol{x}_{c,|\tau|}, \boldsymbol{u}_{c,|\tau|}, t_{|\tau|})\}$, where $\boldsymbol{x}_{c,i}$, and $\boldsymbol{u}_{c,i}$ represent the composite state and control input at time t_i, respectively.

Let $\boldsymbol{C}_\phi(\boldsymbol{x}_c)$ denote a parametric model of the unknown composition matrix, defined in terms of the parameter vector ϕ. Given a collection of pre-trained submodels $PHNN_{i,\Theta}(\cdot)$ for $i = 1, \ldots, k$, the composite model $PHNN^{\boldsymbol{C}_\phi}_{c,\Theta_c}(\cdot)$ is constructed as defined in §5.1, with the exception of the parametrized composition term $\boldsymbol{C}_\phi(\boldsymbol{x}_c)$ being used in place of its ground-truth counterpart.

To learn the parameters ϕ using the dataset $\mathcal{D}_c$, we define a loss function $\mathcal{L}^{comp}(\phi, \Theta_c, \mathcal{D}_c)$ similarly to as in equation 3, where Θ_c denotes the collection of all of the parameter vectors Θ_i from the subsystem PHNNs. In general, $\boldsymbol{C}_\phi(\boldsymbol{x}_c)$ will be a function of $\boldsymbol{x}_c$ and we parametrize its entries as the output of a neural network. $\mathcal{L}^{comp}(\phi, \Theta_c, \mathcal{D}_c)$ can then be minimized by fixing the pre-trained values of Θ_c and performing gradient descent over ϕ.

However, when the interconnection term $\boldsymbol{J}_c(\boldsymbol{x}_c)$ is a constant matrix, $\boldsymbol{C}_\phi$ may be parameterized as constant skew symmetric matrix. In such scenarios, so long as the timestep $t_{i+1} - t_i$ between the recorded datapoints is sufficiently small, each datapoint yields the following collection of linear equations in the unknown entries of $\boldsymbol{C}_\phi$.

$$\frac{\boldsymbol{x}_{i+1} - \boldsymbol{x}_i}{t_{i+1} - t_i} - \left[\tilde{\boldsymbol{J}}_c - \boldsymbol{R}_{c,\theta}(\boldsymbol{x}_{c,i})\right] \nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_{c,i}) + \boldsymbol{G}_{c,\theta}(\boldsymbol{x}_{c,i})\boldsymbol{u}_{c,i} \approx \boldsymbol{C}_\phi \nabla_{\boldsymbol{x}} H_{c,\theta}(\boldsymbol{x}_{c,i}), \quad (4)$$

where we use $\tilde{\boldsymbol{J}}_c$ as shorthand for $\text{Diag}(\boldsymbol{J}_1, \ldots, \boldsymbol{J}_k)$. Dataset $\mathcal{D}_c$ thus yields a least squares problem that we solve to obtain the entries of $\boldsymbol{C}_\phi$.

5.3. Bounding the Errors of Compositions of Port-Hamiltonian Neural Networks

Theorem 1 *Suppose that the true dynamics of each subsystem may be written in port-Hamiltonian form as $P_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$ for $i = 1, 2, \ldots, k$. Furthermore, suppose that the composite system of interest may be represented as a composition $P_c^{\boldsymbol{C}}(\boldsymbol{x}_c, \boldsymbol{u}_c)$ of the port-Hamiltonian subsystems defined by the composition term $\boldsymbol{C}(\boldsymbol{x}_c)$. Let $\boldsymbol{C}_\phi(\boldsymbol{x}_c)$ denote the learned composition term. Suppose that $\|P_i(\boldsymbol{x}_i, \boldsymbol{u}_i) - P_{i,\Theta}(\boldsymbol{x}_i, \boldsymbol{u}_i)\| \leq \varepsilon_i$, $\|\nabla_{\boldsymbol{x}_i} H_i(\boldsymbol{x}_i) - \nabla_{\boldsymbol{x}_i} H_{i,\theta}(\boldsymbol{x}_i)\| \leq \eta_i$, and $\left\|(\boldsymbol{C}(\boldsymbol{x})_{i,j} - \boldsymbol{C}_\phi(\boldsymbol{x})_{i,j}) \nabla_{\boldsymbol{x}_j} H_j(\boldsymbol{x}_j)\right\| \leq \gamma_{i,j}$, for every $\boldsymbol{x}_i \in \Omega_{\boldsymbol{x}_i}$, $\boldsymbol{x}_j \in \Omega_{\boldsymbol{x}_j}$, and $\boldsymbol{u}_i \in \Omega_{\boldsymbol{u}_i}$ for $i = 1, 2, \ldots, k$. Then,*

$$\left\| P_c^{\boldsymbol{C}}(\boldsymbol{x}_c, \boldsymbol{u}_c) - P_{c,\Theta_c}^{\boldsymbol{C}_\phi}(\boldsymbol{x}_c, \boldsymbol{u}_c) \right\| \leq \sum_{i=1}^{k} \left[\varepsilon_i + 2 \sum_{j>i}^{k} (\gamma_{i,j} + \sigma_{i,j} \eta_j) \right], \tag{5}$$

for every $\boldsymbol{x}_c \in \Omega_{\boldsymbol{x}_1} \times \ldots \times \Omega_{\boldsymbol{x}_k}$ and $\boldsymbol{u}_c \in \Omega_{\boldsymbol{u}_1} \times \ldots \times \Omega_{\boldsymbol{u}_k}$. Here, $\sigma_{i,j} := \max_{\boldsymbol{x}_c \in \Omega_{\boldsymbol{x}_c}} \|\boldsymbol{C}_\phi(\boldsymbol{x}_c)_{i,j}\|$ where $\|\boldsymbol{C}_\phi(\boldsymbol{x}_c)_{i,j}\| := \sup\{\|\boldsymbol{C}_\phi(\boldsymbol{x}_c)_{i,j} y\|$ s.t. $y \in \mathbb{R}^{n_j}, \|y\| = 1\}$ is the matrix norm of the submatrix that defines the interactions between subsystems i and j.

In words, Theorem 1 tells us that the prediction error of the composite PHNN is bounded by the error introduced by its component models and the error introduced by the influence that its component models have on each other. The latter is a function of the error in our estimate of the composition term $\boldsymbol{C}_\phi(\boldsymbol{x})$ and in the gradients of the subsystem Hamiltonian functions $\nabla_{\boldsymbol{x}_i} H_{i,\theta}(\boldsymbol{x}_i)$. We note that when $\boldsymbol{C}(\boldsymbol{x})$ is known a priori this result still holds with all $\gamma_{i,j} = 0$. This result ensures that any improvements to the prediction accuracy of the submodels during training will improve the prediction accuracy of the composite model as well. The proof is provided in Appendix A.

6. Numerical Experiments

For detailed descriptions of the dataset generation, the employed neural network architectures, and the training algorithms used to generate the results, we refer the reader to Appendix B. Code to reproduce all experiments is available at github.com/cyrusneary/compositional_port_hamiltonian_NNs.

6.1. Composing Port-Hamiltonian Neural Networks Using a Known Composition Term

We begin by considering a numerical simulation of the coupled spring-mass-dampers illustrated in Figure 2. Recall that this example includes external control forces and nonlinear dissipation.

We first consider the scenario in which the composition term $\boldsymbol{C}$ is known a priori. We learn the subsystem models $PHNN_{1,\Theta}(\cdot)$ and $PHNN_{2,\Theta}(\cdot)$ using separate datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, each of which contains 100 trajectories with randomly sampled initial states. Throughout training we occasionally freeze the network parameters and compose the resulting submodels to obtain a composite model $PHNN_{c,\Theta}^{\boldsymbol{C}}(\cdot)$.

Figure 4 illustrates the result of evaluating the prediction error of both submodels, and of the composite model, on separate testing datasets. The prediction loss measures the average Euclidean distance between the model-predicted future states and the true next states. We plot the median loss values over 10 independent experimental runs, while the shaded regions enclose the 25^{th} and 75^{th} percentiles. Each testing dataset contains 20 system trajectories beginning from randomly sampled initial states, and with different control inputs than the training datasets.

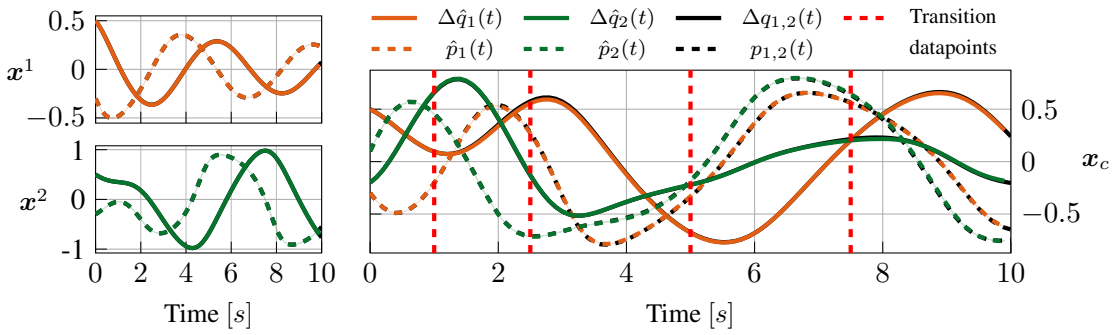

Figure 5: The composite PHNN accurately predicts state trajectories when the unknown composition term $\boldsymbol{C}_\phi$ is inferred using only four datapoints, illustrated by the dotted red lines. Left: Predicted subsystem dynamics. Right: Predicted composite system dynamics.

The composite model accurately predicts dynamics without data from the composite system. We emphasize that the composite model's loss values (blue) in Figure 4 are not the result of training a separate model using composite system data. Instead, the figure shows the result of learning the subsystem models independently (orange and green), and using a physics-informed composition of the submodels to accurately predict the dynamics of a more complex composite system for which we have no data. We additionally observe from Figure 4, that as the prediction losses of the submodels decrease during training, the loss of the composite model decreases as well. This observation empirically demonstrates Theorem 1— the composite model's error should decrease with the error of the subsystem PHNNs.

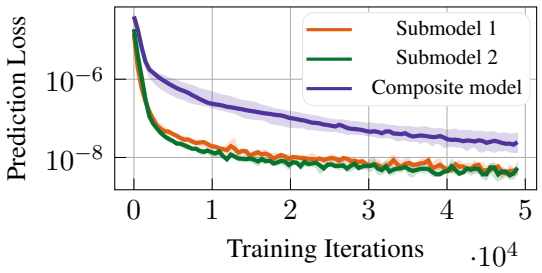

Figure 4: Prediction loss values throughout training. The composite model makes accurate predictions without any training data from the composite system itself.

Finally, we note that training a baseline model for comparison in this example is not possible; the idea of composing deep learning submodels of dynamical systems is novel, and no monolithic baseline model could possibly learn to predict the composite dynamics without training data.

6.2. Learning the Interactions Between Subsystem Port-Hamiltonian Neural Networks

We now proceed to the setting in which the composition term $\boldsymbol{C}$ is unknown—we do not have a priori knowledge of how the subsystems are influencing each other. Instead, as described in §5.2, we assume access to some small dataset of $\mathcal{D}_c$ of observations of the dynamics of the composite system. Specifically, $\mathcal{D}_c$ contains only four datapoints, each of which corresponds to a single-timestep transition beginning from the one of the composite states highlighted by the dotted red lines in Figure 5. Using these datapoints, along with the pretrained submodels $PHNN_{1,\Theta}(\cdot)$ and $PHNN_{2,\Theta}(\cdot)$, we learn $\boldsymbol{C}_\phi$ by solving the least squares problem described in §5.2.

We accurately learn the unknown composition term using a negligible amount of data. Figure 5 illustrates state trajectories predicted by the subsystem, and composite, PHNNs. In all of the subplots, the true dynamics (black) are matched very closely by the PHNN-predicted dynamics (green & orange); we accurately learn the composition term C_ϕ using only four datapoints from the composite system. Monolithic approaches cannot learn effective models from such a limited dataset.

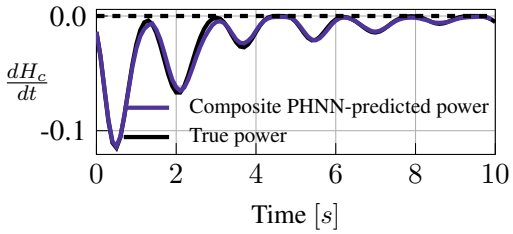

Figure 6: Time derivative of the system's Hamiltonian with no external control inputs.

The learned composite model enjoys properties of port-Hamiltonian systems. Figure 6 illustrates the time derivative of the learned composite Hamiltonian along a predicted trajectory without external control inputs. We note that for the entire trajectory, $dH_{c,\theta}/dt \leq 0$, which provides an empirical demonstration of the cyclo-passivity of the composite PHNN discussed in §4.

6.3. Predicting the Dynamics of Ten Interacting Subsystems Without Additional Training

We now consider ten interacting spring-mass-damper systems, most of which are identical to subsystem 1 from Figure 2, while the rest match subsystem 2. A sinusoidal force acts on one end of the chain. Figure 7 illustrates the results of composing the previously trained PHNN submodels to predict this system's dynamics. The predicted state trajectory of each subsystem is plotted as a separate colored curve.

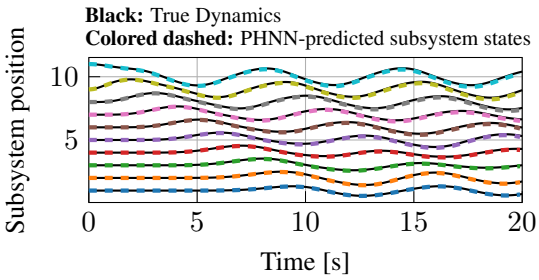

Figure 7: The composite PHNN accurately predicts the dynamics of 10 interacting subsystems with no additional training.

The composite PHNN accurately predicts the wave-like propagation of energy between the subsystems (the colored PHNN predictions overlay the true dynamics in black). We emphasize that this behavior is captured without any additional training of the subsystem models, and without access to any data from the ten-component system; PHNNs can be composed in a modular fashion to simulate entirely new complex systems.

7. Conclusions

In this work we present a framework, algorithms, and theoretical results for the compositional learning of dynamical system models via port-Hamiltonian neural networks (PHNNs). This work presents a first step towards learning modular neural network parametrizations of control systems that can be trained and tested independently, and that can be re-used in new contexts. We demonstrate that by using the structure of port-Hamiltonian systems as inductive bias, we may independently learn submodels on data generated by relatively simple subsystems, and then accurately predict the dynamics of more complex composite systems while using little to no data from the composite system itself. Future work will aim to learn compositional PHNNs from video observations, and also to use the compositional models for control.

References

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Flávio Luiz Cardoso-Ribeiro, Denis Matignon, and Valérie Pommier-Budinger. Modeling of a fluid-structure coupled system using port-hamiltonian formulation. *IFAC-PapersOnLine*, 48(13): 217–222, 2015. 5th IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control LHMNC 2015.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Karim Cherifi, Volker Mehrmann, and Kamel Hariche. Numerical methods to compute a minimal realization of a port-hamiltonian system. *arXiv preprint arXiv:1903.07042*, 2019.

Karim Cherifi, Pawan Kumar Goyal, and Peter Benner. A non-intrusive method to inferring linear port-hamiltonian realizations using time-domain data. *Electronic Transactions on Numerical Analysis: Special Issue SciML*, 56:102–116, 2022.

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

Siyuan Dai, Zhenkai Zhang, and Xenofon Koutsoukos. A model-based design approach for simulation and virtual prototyping of automotive control systems using port-hamiltonian systems. *Software & Systems Modeling*, 18(3):1637–1653, 2019.

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.

Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics and Control Conference*, pages 263–277. PMLR, 2022a.

Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Taylor-lagrange neural ordinary differential equations: Toward fast training and evaluation of neural odes. *arXiv preprint arXiv:2201.05715*, 2022b.

Alejandro Donaire and Sergio Junco. Derivation of input-state-output port-hamiltonian systems from bond graphs. *Simulation Modelling Practice and Theory*, 17(1):137–151, 2009.

Vincent Duindam, Alessandro Macchelli, Stefano Stramigioli, and Herman Bruyninckx. *Modeling and control of complex physical systems: the port-Hamiltonian approach*. Springer Science & Business Media, 2009.

Thai Duong and Nikolay Atanasov. Hamiltonian-based neural ode networks on the se (3) manifold for dynamics learning and control. In *Robotics: Science and Systems (RSS)*, 2021.

Sølve Eidnes, Alexander J Stasik, Camilla Sterud, Eivind Bøhn, and Signe Riemer-Sørensen. Port-hamiltonian neural networks with state dependent ports. *arXiv preprint arXiv:2206.02660*, 2022.

Marc Finzi, Ke Alexander Wang, and Andrew G Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. In *Advances in Neural Information Processing Systems*, volume 33, pages 13880–13889. Curran Associates, Inc., 2020.

Luca Furieri, Clara Lucía Galimberti, Muhammad Zakwan, and Giancarlo Ferrari-Trecate. Distributed neural network control with dependability guarantees: a compositional port-hamiltonian approach. In *Learning for Dynamics and Control Conference*, pages 571–583. PMLR, 2022.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jayesh K Gupta, Kunal Menda, Zachary Manchester, and Mykel Kochenderfer. Structured mechanical models for robot learning and control. In *Learning for Dynamics and Control*, pages 328–337. PMLR, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nicolas Lopes, Thomas Hélie, and Antoine Falaize. Explicit second-order accurate method for the passive guaranteed simulation of port-hamiltonian systems. *IFAC-PapersOnLine*, 48(13):223–228, 2015.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*. OpenReview.net, 2019.

Takashi Matsubara, Ai Ishikawa, and Takaharu Yaguchi. Deep energy-based modeling of discrete-time physics. In *Advances in Neural Information Processing Systems*, volume 33, pages 13100–13111. Curran Associates, Inc., 2020.

Kunal Menda, Jayesh K Gupta, Zachary Manchester, and Mykel J Kochenderfer. Structured mechanical models for efficient reinforcement learning. In *Workshop on Structure and Priors in Reinforcement Learning, International Conference on Learning Representations*, pages 138–171, 2019.

SP Nageshrao, GAD Lopes, D Jeltsema, and R Babuska. Adaptive and learning control of port-hamiltonian systems: A survey. *IEEE Transactions on Automatic Control*, page 37, 2015.

Santiago Sanchez-Escalonilla Plaza, Rodolfo Reyes-Báez, and Bayu Jayawardhana. Total energy shaping with neural interconnection and damping assignment-passivity based control. In *Learning for Dynamics and Control Conference*, pages 520–531. PMLR, 2022.

Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

Ramy Rashad, Davide Bicego, Jelle Zult, Santiago Sanchez-Escalonilla, Ran Jiao, Antonio Franchi, and Stefano Stramigioli. Energy aware impedance control of a flying end-effector in the port-hamiltonian framework. *IEEE transactions on robotics*, 2021.

Manuel A Roehrl, Thomas A Runkler, Veronika Brandtstetter, Michel Tokic, and Stefan Obermayer. Modeling system dynamics with physics-informed neural networks based on lagrangian mechanics. *IFAC-PapersOnLine*, 53(2):9195–9200, 2020.

Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *International Conference on Robotics and Automation*, pages 9784–9790, 2019.

Felix Strehle, Martin Pfeifer, Lukas Kölsch, Charlotte Degünther, Johannes Ruf, Lisa Andresen, and Sören Hohmann. Towards port-hamiltonian modeling of multi-carrier energy systems: A case study for a coupled electricity and gas distribution system. *IFAC-PapersOnLine*, 51(2): 463–468, 2018.

Peter Toth, Danilo J. Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations*. OpenReview.net, 2020.

Arjan Van Der Schaft. *L2-gain and passivity techniques in nonlinear control*. Springer, 2000.

Arjan Van Der Schaft. Port-hamiltonian modeling for control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):393–416, 2020.

Arjan Van Der Schaft, Dimitri Jeltsema, et al. Port-hamiltonian systems theory: An introductory overview. *Foundations and Trends in Systems and Control*, 1(2-3):173–378, 2014.

Liang Xu, Muhammad Zakwan, and Giancarlo Ferrari-Trecate. Neural energy casimir control. *arXiv preprint arXiv:2112.03339*, 2021.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Dissipative symoden: Encoding hamiltonian dynamics with dissipation and control into deep learning. *arXiv preprint arXiv:2002.08860*, 2020.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Benchmarking energy-conserving neural networks for learning dynamics from data. In *Learning for Dynamics and Control*, pages 1218–1229. PMLR, 2021a.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Extending lagrangian and hamiltonian neural networks with differentiable contact models. In *Advances in Neural Information Processing Systems*, volume 34, pages 21910–21922. Curran Associates, Inc., 2021b.

Compositional Learning of Dynamical System Models Using Port-Hamiltonian Neural Networks: Supplementary Material

Appendix A. Proof of Theorem 1

Theorem 1 *Suppose that the true dynamics of each subsystem may be written in port-Hamiltonian form as $P_i(\boldsymbol{x}_i, \boldsymbol{u}_i)$ for $i = 1, 2, \ldots, k$. Furthermore, suppose that the composite system of interest may be represented as a composition $P_c^{\boldsymbol{C}}(\boldsymbol{x}_c, \boldsymbol{u}_c)$ of the port-Hamiltonian subsystems defined by the composition term $\boldsymbol{C}(\boldsymbol{x}_c)$. Let $\boldsymbol{C}_\phi(\boldsymbol{x}_c)$ denote the learned composition term. Suppose that $\|P_i(\boldsymbol{x}_i, \boldsymbol{u}_i) - P_{i,\Theta}(\boldsymbol{x}_i, \boldsymbol{u}_i)\| \leq \varepsilon_i$, $\|\nabla_{\boldsymbol{x}_i} H_i(\boldsymbol{x}_i) - \nabla_{\boldsymbol{x}_i} H_{i,\theta}(\boldsymbol{x}_i)\| \leq \eta_i$, and $\left\|(\boldsymbol{C}(\boldsymbol{x})_{i,j} - \boldsymbol{C}_\phi(\boldsymbol{x})_{i,j}) \nabla_{\boldsymbol{x}_j} H_j(\boldsymbol{x}_j)\right\| \leq \gamma_{i,j}$, for every $\boldsymbol{x}_i \in \Omega_{\boldsymbol{x}_i}$, $\boldsymbol{x}_j \in \Omega_{\boldsymbol{x}_j}$, and $\boldsymbol{u}_i \in \Omega_{\boldsymbol{u}_i}$ for $i = 1, 2, \ldots, k$. Then,*

$$\left\| P_c^{\boldsymbol{C}}(\boldsymbol{x}_c, \boldsymbol{u}_c) - P_{c,\Theta_c}^{\boldsymbol{C}_\phi}(\boldsymbol{x}_c, \boldsymbol{u}_c) \right\| \leq \sum_{i=1}^{k} \left[\varepsilon_i + 2 \sum_{j>i}^{k} (\gamma_{i,j} + \sigma_{i,j} \eta_j) \right], \tag{6}$$

for every $\boldsymbol{x}_c \in \Omega_{\boldsymbol{x}_1} \times \ldots \times \Omega_{\boldsymbol{x}_k}$ and $\boldsymbol{u}_c \in \Omega_{\boldsymbol{u}_1} \times \ldots \times \Omega_{\boldsymbol{u}_k}$. Here, $\sigma_{i,j} := \max_{\boldsymbol{x}_c \in \Omega_{\boldsymbol{x}_c}} \|\boldsymbol{C}_\phi(\boldsymbol{x}_c)_{i,j}\|$ where $\|\boldsymbol{C}_\phi(\boldsymbol{x}_c)_{i,j}\| := \sup\{\|\boldsymbol{C}_\phi(\boldsymbol{x}_c)_{i,j} y\| \text{ s.t. } y \in \mathbb{R}^{n_j}, \|y\| = 1\}$ is the matrix norm of the submatrix that defines the interactions between subsystems i and j.

Proof We begin by expanding the expression for the error in the composite PHNN.

$$\begin{aligned}
&||P_c(\boldsymbol{x}_c, \boldsymbol{u}_c) - P_{c,\Theta_c}^{\boldsymbol{C}_\phi}(\boldsymbol{x}_c, \boldsymbol{u}_c)|| && (7)\\
&\quad = || \left[\tilde{\boldsymbol{J}}_c(\boldsymbol{x}_c) + \boldsymbol{C}(\boldsymbol{x}_c) - \boldsymbol{R}_c(\boldsymbol{x}_c) \right] \nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) + \boldsymbol{G}_c(\boldsymbol{x}_c) \boldsymbol{u}_c && (8)\\
&\qquad - \left[\tilde{\boldsymbol{J}}_c(\boldsymbol{x}_c) + \boldsymbol{C}_\phi(\boldsymbol{x}_c) + \boldsymbol{R}_{c,\theta}(\boldsymbol{x}_c) \right] \nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c) + \boldsymbol{G}_{c,\theta}(\boldsymbol{x}_c) \boldsymbol{u}_c || \\
&\quad \leq || \left[\tilde{\boldsymbol{J}}_c(\boldsymbol{x}_c) - \boldsymbol{R}_c(\boldsymbol{x}_c) \right] \nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) + \boldsymbol{G}_c(\boldsymbol{x}_c) \boldsymbol{u}_c && (9)\\
&\qquad - \left[\tilde{\boldsymbol{J}}_c(\boldsymbol{x}_c) - \boldsymbol{R}_{c,\theta}(\boldsymbol{x}_c) \right] \nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c) - \boldsymbol{G}_{c,\theta}(\boldsymbol{x}_c) \boldsymbol{u}_c || \\
&\qquad + \|\boldsymbol{C}(\boldsymbol{x}_c) \nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) - \boldsymbol{C}_\phi(\boldsymbol{x}_c) \nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c)\| \\
&\quad \leq \sum_{i=1}^{k} \big\{ || \left[\boldsymbol{J}_i(\boldsymbol{x}_i) - \boldsymbol{R}_i(\boldsymbol{x}_i) \right] (\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c))_{\mathcal{I}_i} + \boldsymbol{G}_i(\boldsymbol{x}_i) \boldsymbol{u}_i && (10)\\
&\qquad - \left[\boldsymbol{J}_{i,\theta}(\boldsymbol{x}_i) - \boldsymbol{R}_{i,\theta}(\boldsymbol{x}_i) \right] (\nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c))_{\mathcal{I}_i} + \boldsymbol{G}_{i,\theta}(\boldsymbol{x}_i) \boldsymbol{u}_i || \big\} \\
&\qquad + \|\boldsymbol{C}(\boldsymbol{x}_c) \nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) - \boldsymbol{C}_\phi(\boldsymbol{x}_c) \nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c)\|
\end{aligned}$$

Recall that we define $\tilde{\boldsymbol{J}}_c(\boldsymbol{x}_c) := \text{Diag}(\boldsymbol{J}_1(\boldsymbol{x}_1), \ldots, \boldsymbol{J}_k(\boldsymbol{x}_k))$. Equation 9 follows from equation 8 by the triangle inequality. Equation 10 follows from equation 9 again by the triangle inequality and due to the block-diagonal structure of $\tilde{\boldsymbol{J}}_c(\boldsymbol{x}_c)$, $\boldsymbol{R}_c(\boldsymbol{x}_c)$, and $\boldsymbol{G}_c(\boldsymbol{x}_c)$.

In equation 10 we use $\mathcal{I}_i$ to denote the set of indexes corresponding to subsystem i within the composite vector $\boldsymbol{x}_c$. More precisely, $\mathcal{I}_i := \{j | j > n_1 + \ldots + n_{i-1} \text{ and } j \leq n_1 + \ldots + n_i\}$. We thus use $(\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c))_{\mathcal{I}_i}$ to denote the vector in $\mathbb{R}^{n_i}$ that results from taking the elements from $\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c)$ indexed by $\mathcal{I}_i$ and discarding the rest of the vector. Then, by definition of $H_{\boldsymbol{x}_c}(\boldsymbol{x}_c)$, $\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) = \sum_{j=1}^{k} \nabla_{\boldsymbol{x}_c} H_j(\boldsymbol{x}_j)$. For every $i \neq j$, we have that $(\nabla_{\boldsymbol{x}_c} H_j(\boldsymbol{x}_j))_{\mathcal{I}_i} = \boldsymbol{0} \in \mathbb{R}^{n_i}$

because of our assumption that the subsystem Hamiltonian $H_j(\cdot)$ only depends on $\boldsymbol{x}_j$. Furthermore, using the same reasoning it must be true that $(\nabla_{\boldsymbol{x}_c} H_j(\boldsymbol{x}_j))_{\mathcal{I}_j} = \nabla_{\boldsymbol{x}_j} H_j(\boldsymbol{x}_j)$. So, we conclude that

$$(\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c))_{\mathcal{I}_i} = \sum_{j=1}^{k} (\nabla_{\boldsymbol{x}_c} H_j(\boldsymbol{x}_j))_{\mathcal{I}_i} = \nabla_{\boldsymbol{x}_i} H_i(\boldsymbol{x}_i). \tag{11}$$

By combining equation 11 with equation 10, we have

$$\begin{aligned} ||P_c(\boldsymbol{x}_c, \boldsymbol{u}_c) - P^{\boldsymbol{C}_\phi}_{c,\Theta_c}(\boldsymbol{x}_c, \boldsymbol{u}_c)|| &\leq \sum_{i=1}^{k} ||P_i(\boldsymbol{x}_i, \boldsymbol{u}_i) - P_{i,\Theta}(\boldsymbol{x}_i, \boldsymbol{u}_i)|| \qquad (12) \\ &\quad + \|\boldsymbol{C}(\boldsymbol{x}_c)\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) - \boldsymbol{C}_\phi(\boldsymbol{x}_c)\nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c)\| \\ &\leq \sum_{i=1}^{k} \varepsilon_i + \|\boldsymbol{C}(\boldsymbol{x}_c)\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) - \boldsymbol{C}_\phi(\boldsymbol{x}_c)\nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c)\|\,. \qquad (13) \end{aligned}$$

We bound the final term in the right-hand side of equation 13 as follows.

$$\begin{aligned} &||\boldsymbol{C}(\boldsymbol{x}_c)\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) - \boldsymbol{C}_\phi(\boldsymbol{x}_c)\nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c)|| \qquad (14) \\ &\quad \leq ||(\boldsymbol{C}(\boldsymbol{x}_c) - \boldsymbol{C}_\phi(\boldsymbol{x}_c))\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c)|| + ||\boldsymbol{C}_\phi(\boldsymbol{x}_c)(\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c) - \nabla_{\boldsymbol{x}_c} H_{c,\theta}(\boldsymbol{x}_c))|| \qquad (15) \\ &\quad \leq \sum_{\substack{i=1 \\ j\neq i}}^{k} \big\{ ||(\boldsymbol{C}(\boldsymbol{x}_c)_{\mathcal{I}_i,\mathcal{I}_j} - \boldsymbol{C}_\phi(\boldsymbol{x}_c)_{\mathcal{I}_i,\mathcal{I}_j})\nabla_{\boldsymbol{x}_j} H_j(\boldsymbol{x}_j)|| \qquad (16) \\ &\qquad + ||\boldsymbol{C}_\phi(\boldsymbol{x}_c)_{\mathcal{I}_i,\mathcal{I}_j}(\nabla_{\boldsymbol{x}_j} H_j(\boldsymbol{x}_j) - \nabla_{\boldsymbol{x}_j} H_{j,\theta}(\boldsymbol{x}_j))|| \big\} \qquad (17) \\ &\quad \leq 2\sum_{\substack{i=1 \\ j>i}}^{k} \big\{ \gamma_{i,j} + ||\boldsymbol{C}_\phi(\boldsymbol{x}_c)||\; ||\nabla_{\boldsymbol{x}_j} H_j(\boldsymbol{x}_j) - \nabla_{\boldsymbol{x}_j} H_{j,\theta}(\boldsymbol{x}_j)|| \big\} \qquad (18) \\ &\quad \leq 2\sum_{\substack{i=1 \\ j>i}}^{k} \gamma_{i,j} + \sigma_{i,j}\eta_j \qquad (19) \end{aligned}$$

Equation 15 follows from equation 14 by adding and subtracting $\boldsymbol{C}_\phi(\boldsymbol{x}_c)\nabla_{\boldsymbol{x}_c} H_c(\boldsymbol{x}_c)$, regrouping terms, and applying the triangle inequality. We obtain equation 16 by rewriting the matrix multiplications from equation 15 using the submatrices $\boldsymbol{C}(\boldsymbol{x}_c)_{\mathcal{I}_i,\mathcal{I}_j}$ of $\boldsymbol{C}(\boldsymbol{x}_c)$ (which correspond to the interactions between subsystem i and j). We exclude $\boldsymbol{C}(\boldsymbol{x}_c)_{\mathcal{I}_i,\mathcal{I}_i}$ from the sum in equation 16 because, by definition, $\boldsymbol{C}(\boldsymbol{x}_c)_{\mathcal{I}_i,\mathcal{I}_i} = \boldsymbol{0}$ for all $i = 1, 2, \ldots, k$.

Finally, by combining equation 19 with equation 13, we obtain the desired result.

■

Appendix B. Additional Experimental Details

B.1. Simulation Details

As described in §3.1, the dynamics of the individual spring-mass-damper systems are given by equation 1 with $H_i(\boldsymbol{x}_i) = \frac{\boldsymbol{p}_i^2}{2m_i} + \frac{k\Delta\boldsymbol{q}_i^2}{2}$ and with

$$\boldsymbol{J}_i = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad \boldsymbol{R}_i(\boldsymbol{x}_i) = \begin{bmatrix} 0 & 0 \\ 0 & b_i \frac{\boldsymbol{p}_i^2}{m_i^2} \end{bmatrix}, \quad \boldsymbol{G}_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{20}$$

for each subsystem. Here, the system states are defined as $\boldsymbol{x}_i = (\Delta\boldsymbol{q}_i, \boldsymbol{p}_i)$ for $i = 1, 2$, where $\Delta\boldsymbol{q}_i$ is the elongation of the spring and $\boldsymbol{p}_i$ is the momentum of the mass.

The dynamics of the coupled system are then given by equation 1 where the Hamiltonian $H_c(\boldsymbol{x}_c)$ of the composite system is given by $H_c(\boldsymbol{x}_c) = H_1(\boldsymbol{x}_1) + H_2(\boldsymbol{x}_2)$ and the composite dynamics may be written in the form of equation 1 with

$$\boldsymbol{J}_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}, \quad \boldsymbol{R}_c(\boldsymbol{x}_c) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & b_1 \frac{\boldsymbol{p}_1^2}{m_1^2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b_2 \frac{\boldsymbol{p}_2^2}{m_2^2} \end{bmatrix}, \quad \boldsymbol{G}_c = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}. \tag{21}$$

In our numerical experiments we use a fixed timestep RK4 integration scheme with a timestep of $\Delta t = 0.01$ for all simulations. We define subsystem 1 to have parameters $m_1 = 1.0$, $k_1 = 1.2$, and $b_1 = 1.7$. Meanwhile, we define subsystem 2 to have parameters $m_2 = 1.0$, $k_2 = 1.5$, and $b_{1.7}$. The training datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ correspond to 100 trajectories, of 500 timesteps each, generated by simulating the dynamics from random initial states uniformly sampled from $(\Delta\boldsymbol{q}_i, \boldsymbol{p}_i) \in [-1.0, 1.0]$. When generating dataset $\mathcal{D}_2$, we additionally include a control input force that varies sinusoidally in time. We similarly generate testing datasets by simulating 20 trajectories from each subsystem beginning from initial states randomly sampled from the same distribution. To generate the testing dataset for the composite system $\mathcal{D}_c$, we simulate 20 trajectories of the coupled dynamics from randomly sampled initial states $(\Delta\boldsymbol{q}_1, \boldsymbol{p}_1, \Delta\boldsymbol{q}_2, \boldsymbol{p}_2) \in [-1.0, 1.0, -1.0, 1.0]$ while applying a sinusoidal forcing function to subsystem 2.

B.2. Neural Network Implementations

All numerical experiments were implemented using the python library *Jax* (Bradbury et al., 2018), in order to take advantage of its automatic differentiation and just-in-time compilation features. All experiments were run locally on a desktop computer with a 12^{th} generation Intel i9 CPU, an Nvidia RTX A4000 GPU, and with 32 GB of RAM.

The Hamiltonian $H_\theta(\boldsymbol{x})$ and dissipation $\boldsymbol{R}_\theta(\boldsymbol{x})$ neural networks are implemented as multilayer perceptrons with TANH activation functions and with two hidden layers of 32 units each. Meanwhile, we parametrize the entries of the constant control input matrix $\boldsymbol{G}_\theta(\boldsymbol{x})$ directly.

We train all models using ADAM (Kingma and Ba, 2014) for a fixed number of training steps with a learning rate of $\alpha = 10^{-3}$. We use a minibatch size of 32. We do not include any additional regularization terms in the loss function.