# Distributed neural network control with dependability guarantees: a compositional port-Hamiltonian approach

**Luca Furieri** LUCA.FURIERI@EPFL.CH
*Institute of Mechanical Engineering, École Polytechnique Fédérale de Lausanne, Switzerland*

**Clara Lucía Galimberti** CLARA.GALIMBERTI@EPFL.CH
*Institute of Mechanical Engineering, École Polytechnique Fédérale de Lausanne, Switzerland*

**Muhammad Zakwan** MUHAMMAD.ZAKWAN@EPFL.CH
*Institute of Mechanical Engineering, École Polytechnique Fédérale de Lausanne, Switzerland*

**Giancarlo Ferrari-Trecate** GIANCARLO.FERRARITRECATE@EPFL.CH
*Institute of Mechanical Engineering, École Polytechnique Fédérale de Lausanne, Switzerland*

## Abstract

[1]Large-scale cyber-physical systems require that control policies are *distributed*, that is, that they only rely on local real-time measurements and communication with neighboring agents. Optimal Distributed Control (ODC) problems are, however, highly intractable even in seemingly simple cases. Recent work has thus proposed training Neural Network (NN) distributed controllers. A main challenge of NN controllers is that they are not *dependable* during and after training, that is, the closed-loop system may be unstable, and the training may fail due to vanishing gradients. In this paper, we address these issues for networks of nonlinear port-Hamiltonian (pH) systems, whose modeling power ranges from energy systems to non-holonomic vehicles and chemical reactions. Specifically, we embrace the compositional properties of pH systems to characterize deep Hamiltonian control policies with *built-in* closed-loop stability guarantees — irrespective of the interconnection topology and the chosen NN parameters. Furthermore, our setup enables leveraging recent results on well-behaved neural ODEs to prevent the phenomenon of vanishing gradients by design. Numerical experiments corroborate the dependability of the proposed architecture, while matching the performance of general neural network policies.

**Keywords:** optimal distributed control, deep learning, port-Hamiltonian systems, neural ODEs

## 1. Introduction

The design of distributed control policies — i.e., policies that can only observe local sensor measurements in real-time — can be extremely challenging even in simple cases. Specifically, Witsenhausen's counter-example (Witsenhausen, 1968) has shown that even when the system dynamics are *linear*, the loss function to minimize is *quadratic*, and the additive noise is *Gaussian*, a *nonlinear* distributed control policy can outperform the best linear one. A body of work culminating in Rotkowitz and Lall (2006); Lessard and Lall (2011) has identified sufficient and necessary conditions — commonly known as Quadratic Invariance (QI) — under which optimal distributed control of linear systems is a convex (i.e., tractable) optimization program. Nonetheless, most large-scale systems are nonlinear, and they violate the requirement of QI due to geographical distance or privacy concerns. These limitations motivate going beyond linear control and suggest parametrizing

1. The code for numerical examples is available at https://github.com/DecodEPFL/DeepDisCoPH.

highly nonlinear distributed policies through Deep Neural Networks (DNNs). Specifically, the recent works Tolstaya et al. (2020); Khan et al. (2020); Gama and Sojoudi (2021); Yang and Matni (2021) have focused on training Graph Neural Networks (GNNs) that parametrize static and dynamical distributed control policies. These methods achieve remarkable performance in applications such as vehicle flocking and formation flying. Furthermore, the special structure of GNNs allows for scalability to very large-scale systems.

The first challenge of using general GNNs as control policies is that it is often possible to guarantee closed-loop stability of the learned policy only under the assumptions that 1) the system dynamics are linear and open-loop stable, and 2) the DNN weights result in small-enough Lipschitz constants across the layers (e.g., Gama and Sojoudi (2021)). As such conditions may not be fulfilled during exploration, the system may incur failures before an optimal policy can be learned (Brunke et al., 2021; Cheng et al., 2019). Recent mitigation strategies include Berkenkamp et al. (2018); Richards et al. (2018); Koller et al. (2018), where the authors propose to improve an initial known safe policy iteratively, while imposing the constraint that the initial *region of attraction* does not shrink. An alternative approach is to exploit integral quadratic constraints to enforce closed-loop stability of DNN controllers for linear systems (Pauli et al., 2021). The main obstacle of these methods is that explicitly constraining the DNN weights may be infeasible or become a bottleneck for closed-loop performance. Recent work proposes stable-by-design control policies based on mechanical energy conservation. However, the corresponding methods are specific to a class of robotic systems (Abdulkhader et al., 2021) and SE(3) dynamics (Duong and Atanasov, 2021).

The second challenge of general $N$-layered GNN policies is that they must be applied recursively at every time step $t = 1, \dots, T$. The corresponding optimal control problem is thus equivalent to a very deep learning problem involving $T \times N$ computational layers in total, which may exacerbate the phenomena of *vanishing and exploding gradients*. Vanishing or exploding gradients may lead to numerical ill-posedness and impede the DNN training from learning a stabilizing controller. Practical methods to mitigate exploding/vanishing gradients include clipping (Goodfellow et al., 2016) and long short-term memory layers (Hochreiter and Schmidhuber, 1997; Pascanu et al., 2013). A different approach is to design DNN architectures that avoid exploding/vanishing gradients *by design*. For instance, Arjovsky et al. (2016); Vorontsov et al. (2017); Helfrich et al. (2018); Choromanski et al. (2020) exploit unitary and orthogonal matrix trainable parameters. These approaches, however, require expensive computations during training (Choromanski et al., 2020), or use restricted classes of weight matrices (Arjovsky et al., 2016; Vorontsov et al., 2017; Helfrich et al., 2018), which may severely limit the expressivity of the models — especially so if combined with constraints for guaranteeing closed-loop stability.

In this paper, we focus our attention on large-scale networks of nonlinear port-Hamiltonian (pH) systems. Many large-scale engineering applications, such as voltage and frequency control in islanded AC microgrids (Strehle et al., 2021), swarms of robots (Angerer et al., 2017), and UAVs (Fahmi and Woolsey, 2021; Muñoz et al., 2013; Mersha et al., 2011), can be modeled as networks of coupled pH systems. Classical methods for stabilization of pH systems around desired equilibria include Passivity-Based Control (PBC) and Control by Interconnection (CbI) (Ortega et al., 2008; Van der Schaft, 2000), and energy-based control via Casimir functions and damping assignment, e.g., Guo and Cheng (2006). When it comes to minimizing a cost function, however, the problem becomes highly intractable. Therefore, only a few works address optimal control of nonlinear pH systems. For instance, Sprangers et al. (2014) use reinforcement learning to include performance considerations into PBC, and Kölsch et al. (2021) incorporate an adaptive feedback component

that achieves optimality for some classes of pH systems and controller architectures. However, it has remained largely unexplored how to design distributed and DNN control policies that preserve closed-loop stability and good numerical properties during training.

**Contributions** Our first result is to parametrize a class of distributed DNN control policies for large networks of pH systems that offers *built-in guarantees* of closed-loop stability and non-vanishing gradients. We then establish sparsity conditions on the trainable matrix weights of our controllers that make policies comply with pre-defined information constraints typical of large-scale applications. We achieve closed-loop stability — irrespective of the network topology and the choice of weights during exploration — by endowing Hamiltonian Deep Neural Networks (H-DNNs) from Galimberti et al. (2021b,a) with input-output ports and exploiting compositional properties of pH systems. As a corollary of Galimberti et al. (2021a), we last show that the corresponding deep learning problem is not affected by the phenomenon of vanishing gradients when the pH dynamics are not dissipative. We validate our results on a multi-robot collision avoidance task and we achieve the performance of general NN control policies. To the best of our knowledge, no previous work offers guarantees of closed-loop and numerical stability for distributed DNN control policies that hold irrespectively of the chosen weights and for a rich class of linear and nonlinear systems.

**Notation** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with nodes $\mathcal{V} = \{1, \dots, M\}$ and edges $\mathcal{E}$, and let $S \in \{0,1\}^{M \times M}$ be the associated adjacency matrix. The set of $k$-hop neighbors of node $i$ with respect to graph $\mathcal{G}$ is denoted as $\mathcal{N}_i^k$, that is, $\mathcal{N}_i^k = \{j \in \mathcal{V} |\ \ S^k(i,j) \neq 0\}$, where $S^k(i,j)$ denotes the entry $(i,j)$ of the matrix $S^k$. In the paper, we associate vectors $v_i \in \mathbb{R}^{n_i}$ with each node of graph $\mathcal{G}$. The set of vectors associated with the $k$-hop neighbors of node $i$ with respect to graph $\mathcal{G}$ is denoted as $\mathbf{v}_i^k = \{v_j |\ \ j \in \mathcal{N}_i^k\}$. For a block-matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, where $m = \sum_{i=1}^{M} m_i$ and $n = \sum_{i=1}^{N} n_i$, we denote its block in position $(i,j)$ as $W_{ij} \in \mathbb{R}^{M_i \times n_i}$. For a binary matrix $R \in \{0,1\}^{M \times N}$, we write that $\mathbf{W} \in \mathrm{blkSparse}(R)$ if and only if $R(i,j) = 0 \implies W_{ij} = 0$. We denote with $\mathbf{1}$ the vector of ones of appropriate dimensions.

## 2. Problem Statement

We consider a network of $M \in \mathbb{N}$ nonlinear dynamically coupled systems, each endowed with a local feedback control policy. Letting $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$ denote the graph associated with the dynamical couplings between systems, and $S_d \in \{0,1\}^{M \times M}$ with $S_d(i,i) = 1$ for all $i = 1, \dots, M$ be the adjacency matrix of $\mathcal{G}_d$, we have

$$\begin{aligned} \dot{x}_i(t) &= f_i(\mathbf{x}_i^1(t), u_i(t))\,, \\ y_i(t) &= h_i(x_i(t))\,, \qquad \forall i = 1, \dots, M\,, \end{aligned} \tag{1}$$

where $x_i(t) \in \mathbb{R}^{n_i}$ is the local system state, $u_i(t) \in \mathbb{R}^{m_i}$ is the local control input, $y_i(t) \in \mathbb{R}^{p_i}$ is the local output, and $\mathbf{x}_i^1(t)$ denotes the set of states associated with 1-hop neighbors of system $i$ according to $\mathcal{G}_d$. When the number $M$ of systems is very large, we say that system (1) is *large-scale*. The main challenge of control for large-scale systems is that the local control inputs $u_i(t)$ can only receive real-time information from a limited subset of neighboring systems according to a communication topology encoded by a graph $\mathcal{G}_c$ with adjacency matrix $S_c \in \{0,1\}^{M \times M}$ such that $S_c(i,i) = 1$ for every $i = 1, \dots, M$. These policies are called *distributed* due to the presence of information constraints. More formally, the goal of ODC is to compute optimal distributed feedback

policies $(\chi_i(\cdot), \pi_i(\cdot))$ such that

$$\dot{\xi}_i(t) = \chi_i(\boldsymbol{\xi}_i^{R_\xi}(t), \mathbf{y}_i^{R_y}(t), t)\,, \tag{2}$$

$$u_i(t) = \pi_i(\boldsymbol{\xi}_i^{R_\xi}(t), \mathbf{y}_i^{R_y}(t), t)\,, \quad \forall i = 1, \dots, M\,, \tag{3}$$

where $\boldsymbol{\xi}(t) = (\xi_1(t), \dots, \xi_M(t))$, and $\mathbf{y}_i^{R_y}(t)$ and $\boldsymbol{\xi}_i^{R_\xi}(t)$ are defined in terms of the communication graph $\mathcal{G}_c$. In (2)-(3) the variable $\xi_i(t) \in \mathbb{R}^{q_i}$ is the internal dynamical state of the controller which acts as *local memory*. In control-theoretic terms, (2)-(3) is a dynamical controller. The value $R_y \in \mathbb{N}$ is the *output measurement radius*, which indicates how far into the graph $\mathcal{G}_c$ local controllers are able to measure the outputs from other systems in real-time. The value $R_\xi \in \mathbb{N}$, instead, is the *communication radius*, which indicates how far into the graph $\mathcal{G}_c$ local controllers are able to exchange their local states $\xi_i(t)$ through communication in real-time.

The policies (2)-(3) should be *optimal*, in the sense that it minimize a loss function

$$\mathcal{L} = \frac{1}{T}\int_0^T l(\mathbf{x}(t), \mathbf{u}(t), t)dt\,, \tag{4}$$

in a finite-horizon $T \in \mathbb{R}$, where $\mathcal{L}$ is differentiable almost everywhere, $\mathbf{x}(t) = (x_1(t), \dots, x_M(t))$ and $\mathbf{u}(t) = (u_1(t), \dots, u_M(t))$.[2] We assume that $(\chi_i(\cdot), \pi_i(\cdot))$ are parametrized through tunable parameters $\vartheta_i(t) \in \mathbb{R}^{d_i}$ for every $t \in \mathbb{R}$ and $i = 1, \dots, M$. Therefore, the ODC amounts to finding optimal parameters $\boldsymbol{\vartheta}(t) = (\vartheta_1(t), \dots, \vartheta_M(t))$ that solve the following optimization program:

$$\min_{\boldsymbol{\vartheta}_{[0,T]}} \quad \mathcal{L}(\boldsymbol{\vartheta}_{[0,T]}) \tag{5}$$

$$\text{s.t.} \quad \text{system dynamics (1)}\,,$$

$$\dot{\xi}_i(t) = \chi_i(\mathbf{y}_i^{R_y}(t), \boldsymbol{\xi}_i^{R_\xi}(t), \vartheta_i(t))\,, \tag{6}$$

$$u_i(t) = \pi_i(\mathbf{y}_i^{R_y}(t), \boldsymbol{\xi}_i^{R_\xi}(t), \vartheta_i(t))\,, \quad \forall i = 1, \dots, M\,, \tag{7}$$

where $\vartheta_{[0,T]}$ denotes the trajectory $\vartheta(t)$ for $t \in [0, T]$. When $l(\cdot)$ is quadratic in $\mathbf{x}(t)$ and $\mathbf{u}(t)$, the functions $f_i(\cdot)$ in (1) are linear in their arguments, and $R_y = R_\xi = M$, linear policies are optimal and (5) can be solved through convex or dynamic programming — this is the standard centralized LQ output-feedback problem. A similar result holds if $R_y$ and $R_\xi$ are strictly smaller than $M$, and the communication $\mathcal{G}_c$ is QI with respect to the dynamical couplings graph $\mathcal{G}_d$ of (1) (Rotkowitz and Lall, 2006). Unfortunately, these assumptions do not hold for the vast majority of real-world large-scale systems. This fact motivates parametrizing the functions $\chi_i(\cdot), \pi_i(\cdot)$ as highly nonlinear DNNs, even when the dynamics (1) are linear (Gama and Sojoudi, 2021; Yang and Matni, 2021).

However, the closed-loop interconnection of the system (1) with the controller in the form (6)-(7) can lead to instability — that is, even if the cost is optimized for a finite-horizon $T$, the infinite-horizon cost is infinite, leading to system failure and safety concerns. Further, the corresponding learning problem may be ill-posed due to vanishing gradients when the horizon $T$ is large. In this work, we overcome these challenges for nonlinear system dynamics (1) that are in pH form.

2. The function $l(\cdot)$ may include an *expected value* over a distribution on $\mathbf{x}(0)$. Our theoretical results are independent of this choice.

### 2.1. Networks of Port-Hamiltonian Systems

A controlled network of nonlinear pH systems is a particular case of (1) with

$$\dot{x}_i(t) = (\Omega_i - R_i)\frac{\partial V_i(x_i(t))}{\partial x_i} + \sum_{j \in \mathcal{N}_i^1 \setminus \{i\}} F_{ij} y_j(t) + G_i^\mathsf{T} u_i(t)\,, \tag{8}$$
$$y_i(t) = G_i \frac{\partial V_i(x_i(t))}{\partial x_i}\,, \quad \forall i = 1, \dots, M\,,$$

where "\" denotes the set difference, $m_i = p_i$ for all $i = 1, \dots, M$, neighbors $j \in \mathcal{N}_i^1$ are defined in terms of the dynamical couplings graph $\mathcal{G}_d$, and it holds that $\Omega_i^\mathsf{T} + \Omega_i = 0$, $R_i \succeq 0$. The functions $V_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}$ are the local *Hamiltonians* of the systems and they describe their internal energy at any time $t \in \mathbb{R}$. We make the following standard assumptions on the Hamiltonian functions:

**A1** $V_i(\cdot)$ is continuously differentiable.

**A2** $\lim_{\|x\| \to \infty} V_i(x) = \infty$, that is, $V_i(\cdot)$ is radially unbounded.

The networked system (8) can be compactly written as

$$\dot{\mathbf{x}}(t) = (\mathbf{\Omega} - \mathbf{R})\frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}} + \mathbf{F}\mathbf{y}(t) + \mathbf{G}^\mathsf{T}\mathbf{u}(t)\,, \tag{9}$$
$$\mathbf{y}(t) = \mathbf{G}\frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}}\,, \tag{10}$$

where $\mathbf{\Omega} = \text{blkdiag}(\Omega_i)$, $\mathbf{R} = \text{blkdiag}(R_i)$, $\mathbf{F} \in \text{blkSparse}(S_d)$ with $F_{ii} = 0$ for every $i = 1, \dots, M$, $\mathbf{G} = \text{blkdiag}(G_i)$ and $V(\mathbf{x}(t)) = \sum_{i=1}^M V_i(x_i(t))$. When the interconnection matrix $\mathbf{FG} \in \text{blkSparse}(S_d)$ is skew-symmetric — implying that if $F_{ij} \neq 0$ then $F_{ji} \neq 0$ for any systems $i$ and $j$ — networks of pH systems in the form (8) possess a *compositional property*, that is, their interconnection preserves stability properties of the local systems (van der Schaft, 2004).

**Proposition 1 (Compositional principle for pH systems)** *Consider the network of pH systems in* (9)-(10) *with no external input* $\mathbf{u}(t) = 0$*, and assume that* $\mathbf{FG} \in \text{blkSparse}(S_d)$ *is skew-symmetric. Then, we have that*

$$\forall x_i(0) \in \mathbb{R}^{n_i}\,, \quad \lim_{t \to \infty} \left( \inf_{\overline{x}_i \in \mathcal{M}_i} \|x_i(t) - \overline{x}_i\| \right) = 0\,, \quad \forall i = 1, \dots, M\,, \tag{11}$$

*where* $\mathcal{M}_i$ *is the largest positively invariant set contained in the set*

$$E_i = \left\{ \overline{x}_i |\ \frac{\partial V_i(\overline{x}_i)}{\partial x_i} \in \text{Ker}(R_i) \right\}\,. \tag{12}$$

We report the proof of Proposition 1 in Appendix A. In plain words, Proposition 1 ensures that *stability properties of pH systems are not compromised by interconnecting arbitrarily many of them* through power-preserving links. Indeed, the states $x_i(t)$ will globally converge to the set $E_i$ — whose definition only depends on $V_i(\cdot)$ and $R_i$ — irrespective of dynamical couplings with neighboring pH systems $j \in \mathcal{N}_i^1$. Dynamical couplings can only affect the invariant sets $\mathcal{M}_i \subseteq E_i$ where the local state variables $x_i(t)$ converge to. We remark that while the result of Proposition 1 holds with full generality, characterizing the invariant sets $\mathcal{M}_i$ requires explicit case-by-case analysis. For example, Proposition 1 may also implies global asymptotic stability, as we showcase in the following example.

**Example** If $R_i \succ 0$ for all $i = 1, \dots, M$, and the Hamiltonians $V_i(\cdot)$ are strongly convex, then $x_i(t)$ globally asymptotically converges to the unique stationary point $x_i^\star$ of $V_i(\cdot)$ for every $i = 1, \dots, M$, irrespective of the network topology and the dynamical couplings with other systems. This is because all $E_i$'s are singletons, and therefore, they are positively invariant.

The compositional principle of Proposition 1 inspires us to parametrize distributed DNN control policies that cannot compromise the stability properties of an arbitrarily large network of pH systems.

## 3. Deep Distributed Control for Port-Hamiltonian Systems (DeepDisCoPH)

We consider time-varying, dynamical control policies in the form

$$\dot{\boldsymbol{\xi}}(t) = (\mathbf{J} - \boldsymbol{R}_c)\frac{\partial \Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\xi}} + \mathbf{K}\mathbf{y}(t)\,, \tag{13}$$

$$\mathbf{u}(t) = -\mathbf{K}^\mathsf{T}\frac{\partial \Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\xi}}\,, \tag{14}$$

where explicit dependence on time is due to $\boldsymbol{\theta}(t)$. In the equation above, we assume that $\mathbf{J}^\mathsf{T} + \mathbf{J} = 0$ and $\mathbf{R}_c \succeq 0$, where all trainable parameters are indicated in **blue color**. The expressivity of control policies in the form (13)-(14) is linked to the scalar function $\Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}(t))$, which parametrizes through $\boldsymbol{\theta}(t)$ the *internal energy* of the controller at any given time. We do not pose any restriction on the function $\Phi(\cdot)$ except from assumptions **A1** and **A2**; for instance, $\Phi(\cdot)$ can be chosen as a simple quadratic function, as a Multi-Layer-Perceptron (MLP), or as a deep H-DNN (Galimberti et al., 2021a). The theoretical results of this paper hold irrespective of this choice.

The class of policies (13)-(14) is chosen to structurally preserve the port-Hamiltonian form in the closed-loop. Our first observation is that, when the trainable parameters of the energy $\Phi(\cdot)$ do not depend on time, i.e.,

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}\,, \quad \forall t \in \mathbb{R}\,, \tag{15}$$

we can apply the compositional property of pH systems and achieve stability of the closed-loop (9)-(10)-(13)-(14) by design — i.e., irrespective of the specific values of $(\mathbf{J}, \mathbf{R}_c, \mathbf{K}, \boldsymbol{\theta})$.

**Corollary 2 (Closed-loop stability by design)** *Consider the pH system* (9)-(10) *with skew-symmetric* $\mathbf{FG}$ *controlled through* (13)-(14). *Assume that* (15) *holds. Then, for all* $x_i(0)$ *and all* $i = 1, \dots, M$, *the local state* $x_i(t)$ *converges to the largest invariant set* $\mathcal{M}_i$ *contained in* $E_i$, *where* $E_i$ *is defined in* (12), *and for all* $\boldsymbol{\xi}(0)$, $\boldsymbol{\xi}(t)$ *converges to the largest invariant set in* $\left\{\overline{\boldsymbol{\xi}} \middle| \frac{\partial \Phi(\overline{\boldsymbol{\xi}}, \boldsymbol{\theta})}{\partial \boldsymbol{\xi}} \in \mathrm{Ker}(\mathbf{R}_c)\right\}$.

The proof is equivalent to Proposition 1 by noticing that the system (9)-(10) augmented with (13)-(14) is Hamiltonian with total energy $\sum_{i=1}^M V_i(x_i(t)) + \Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta})$. We remark that the time-invariant assumption (15) is only needed to predict the infinite-time behavior of the closed-loop system. We do not require (15) during policy training because the performance metric $\mathcal{L}(\boldsymbol{\theta}_{[0,T]})$ is defined in a finite-horizon. Hence, we suggest training a time-varying policy (13)-(14) for the time horizon of interest $[0, T]$ in order to preserve maximal expressivity, and then freeze the parameters $\boldsymbol{\theta}(t) = \boldsymbol{\theta}(T)$ for all $t > T$ in order to preserve closed-loop stability according to Corollary 2.

The control policy (13)-(14) is *not distributed*, in general. Indeed, the computation of a local control input $u_i(t)$ for some $i \in \{1, \dots, M\}$ may involve output measurements $y_j(t)$ and controller

internal states $\xi_k(t)$ of all the other systems. Our next result is to establish structural conditions on the trainable parameters and the energy function $\Phi(\cdot)$ to enforce a desired output measurement radius $R_y \in \mathbb{N}$ and communication radius $R_\xi \in \mathbb{N}$. The proof of Theorem 3 is reported in Appendix B.

**Theorem 3 (Distributed H-DNN controllers)** *Consider the pH system (9)-(10) with* **FG** *skew-symmetric controlled through (13)-(14). Assume that, for all $t \in \mathbb{R}$ and some $L_y, L_\xi \in \mathbb{N}$:*

**D1** *the matrices* $\mathbf{J}$, $\mathbf{R}_c$ *and* $\mathbf{K}$ *all belong to* $\mathrm{blkSparse}((S_c)^{L_y})$,

**D2** *the controller energy $\Phi(\cdot)$ is* separable *as per*

$$\Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}(t)) = \sum_{i=1}^{M} \Phi_i(\boldsymbol{\xi}_i^{L_\xi}(t), \theta_i(t)) \,, \tag{16}$$

*that is, it is the sum of $M$ local energies, each one defined as a function of the internal state variables of the $L_\xi$-hop neighboring controllers according to the communication graph $\mathcal{G}_c$.*

*Then, the control policy (13)-(14) is distributed with output measurement radius $R_y$ and communication radius $R_\xi$ if*

$$L_y \leq R_y, \text{ and } \quad L_y + 2L_\xi \leq R_\xi \,. \tag{17}$$

**Example** Consider the scenario where $R_y = R_\xi = 1$, corresponding to controllers that can only measure outputs and exchange information with their one-hop neighbors according to the communication graph $\mathcal{G}_c$. We let $L_y = 1$ and $L_\xi = 0$ to comply with the condition (17). Then, the assumption **D1** is satisfied by letting matrices $\mathbf{J}$, $\mathbf{R}_c$ and $\mathbf{K}$ all lie in the subspace $\mathrm{blkSparse}(S_c)$. We satisfy assumption **D2** by letting the total controller energy $\Phi(\cdot)$ be completely separable as per

$$\Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}(t)) = \sum_{i=1}^{M} \Phi_i(\xi_i(t), \theta_i(t)) \,.$$

For instance, letting $w_i$ be a trainable row vector, one may choose the local energies as

$$\Phi_i(\xi_i(t), \theta_i) = w_i \sigma(W_{i,2} \sigma(W_{i,1} \xi_i(t))) \,, \quad \forall i = 1, \ldots, M \,, \tag{18}$$

which corresponds to the output of a two-layer NN with activation function $\sigma(\cdot)$.

## 4. Training Distributed H-DNN Controllers Beyond Vanishing Gradients

Having established a class of distributed control policies that preserves closed-loop stability by design, we now seek a low-cost stationary point for (5) by training a Neural ODE. For details on its implementation, we refer the interested reader to Chen et al. (2018). In summary, neural ODEs are neural network models that generalize standard layer to layer propagation to continuous-time dynamics. This is achieved by interpreting the function $\mathbf{f}(\cdot)$ of an ODE $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}(t))$ as a NN endowed with an integration method. We remark that discretization during training can introduce suboptimality, but does not compromise the closed-loop stability result of Corollary 2; indeed, stability of the continuous-time systems holds irrespectively of the chosen weights.

Next, we derive the neural ODE model associated with the trainable closed-loop system (9)-(10) controlled through (13)-(14). Then, we establish a property of non-vanishing gradients during training for non-dissipative systems — irrespective of the neural ODE depth.

**ODC as an H-DNN** The neural ODE associated with our class of pH systems and controllers is expressed as

$$\dot{\boldsymbol{\zeta}}(t) = \begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\boldsymbol{\xi}}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Omega} + \mathbf{FG} - \mathbf{R} & -\mathbf{G}^\mathsf{T}\mathbf{K}^\mathsf{T} \\ \mathbf{KG} & (\mathbf{J} - \mathbf{R}_c) \end{bmatrix} \begin{bmatrix} \frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}} \\ \frac{\partial \Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\xi}} \end{bmatrix} \tag{19}$$

$$= (\boldsymbol{\Psi} - \mathbf{S}) \frac{\partial P(\boldsymbol{\zeta}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\zeta}}, \tag{20}$$

where $\boldsymbol{\Psi}$ is skew-symmetric and $\mathbf{S} \succeq 0$ for every $t \in \mathbb{R}$.[3] The Hamiltonian function of the closed-loop system is expressed as $P(\mathbf{x}(t), \boldsymbol{\xi}(t), \boldsymbol{\theta}(t)) = V(\mathbf{x}(t)) + \Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}(t))$. The corresponding neural ODE has the form of a time-varying Hamiltonian system — as recently proposed in Galimberti et al. (2021a). For instance, upon implementing a Forward Euler (FE) discretization scheme for (20), the corresponding neural network architecture is akin to $\mathrm{H}_1$-DNN from Galimberti et al. (2021a). Here, the main difference with respect to Galimberti et al. (2021a) is that only the controller parameters highlighted in blue in (19) can be trained through back-propagation, and that the performance metric considers the entire trajectory $\boldsymbol{\zeta}_{[0,T]}$ rather than $\boldsymbol{\zeta}(T)$ only. Further, the energy $\Phi(\cdot)$ is not restricted to a fixed model, but can be chosen arbitrarily, e.g. (18). Irrespective of this choice, we show that the Backward Sensitivity Matrices (BSMs) cannot vanish.

**Corollary 4 (of Theorem 1 in Galimberti et al. (2021a))** *Consider the ODC problem (5). Assume that the networked system and the local control policies are in pH form, i.e. (9)-(10) with* $\mathbf{FG}$ *skew-symmetric and (13)-(14) hold. Further assume that* $\boldsymbol{R} = \boldsymbol{R}_c = 0$*. Then,*

$$\left\| \frac{\partial \boldsymbol{\zeta}(\tilde{t})}{\partial \boldsymbol{\zeta}(\tilde{t} - t)} \right\| \geq 1, \quad \forall t, \tilde{t}, T \in \mathbb{R}, \text{ such that } t \leq \tilde{t} \leq T,$$

*where* $\boldsymbol{\zeta}(t) = (\mathbf{x}(t), \boldsymbol{\xi}(t))$*, and* $\|\cdot\|$ *is any sub-multiplicative norm with* $\left\|X^\mathsf{T}\right\| = \|X\|$.

The proof of Corollary 4 is provided in Appendix C. As it was shown in Galimberti et al. (2021a), the fact that the BSM $\frac{\partial \boldsymbol{\zeta}(\tilde{t})}{\partial \boldsymbol{\zeta}(\tilde{t}-t)}$ cannot vanish as $(\tilde{t}-t)$ increases plays a key role in guaranteeing non-vanishing gradients for arbitrarily deep H-DNN implementations of the ODC (5). Indeed, letting $\theta_{i,j}$ denote the scalar parameter $i$ of layer $j$ of the $N$-layered neural ODE implementation, the backward propagation algorithm performs the following computations

$$\frac{\partial \mathcal{L}}{\partial \theta_{i,j}} = \frac{\partial \boldsymbol{\zeta}_{j+1}}{\partial \theta_{i,j}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\zeta}_{j+1}} = \frac{\partial \boldsymbol{\zeta}_{j+1}}{\partial \theta_{i,j}} \sum_{k=j+1}^{N} \left[ \left( \frac{\partial \boldsymbol{\zeta}_k}{\partial \boldsymbol{\zeta}_{j+1}} \right) \frac{\partial \mathcal{L}}{\partial \boldsymbol{\zeta}_k} \right]. \tag{21}$$

Vanishing (resp., exploding) gradients are linked to the term $\frac{\partial \boldsymbol{\zeta}_k}{\partial \boldsymbol{\zeta}_{j+1}}$ vanishing to zero (resp., diverging to infinity) as $k \to N$ and $N \to \infty$. Since $\frac{\partial \boldsymbol{\zeta}_k}{\partial \boldsymbol{\zeta}_{j+1}}$ is the discrete-time counterpart of $\frac{\partial \boldsymbol{\zeta}(\tilde{t})}{\partial \boldsymbol{\zeta}(\tilde{t}-t)}$, our result of Corollary 4 ensures that the proposed architecture prevents vanishing gradients in continuous-time, and may only appear due to 1) an excessively large discretization step in the neural ODE implementation, or 2) the resistive part of pH systems.

3. Note that both $\boldsymbol{\Psi}$ and $\mathbf{S}$ contain both trainable (in **blue**) and non-trainable (in **black**) parameters as per (19). We choose nevertheless to represent them in **blue**.

## 5. Numerical experiments

In this section, we validate our results on dependability and performance of H-DNN control policies on an ODC task. We consider a fleet of $M = 12$ mobile robots that need to achieve a formation on the $xy$-plane within a given time $T \in \mathbb{R}$. The robots can only rely on neighbors' information, must avoid collisions between each other, and their trajectories should minimize a given loss function. Each robot $i = 1, \dots, M$ is modeled through point-mass dynamics with mass $m_i > 0$, and the control input $\mathbf{u}(t) = (u_1(t), \dots, u_M(t))$ is the force that local motors can apply to the corresponding robot. The state of robot $i$ is given by $x_i(t) = (q_{i,x}(t), q_{i,y}(t), p_{i,x}(t), p_{i,y}(t))$, where $(q_{i,x}(t), q_{i,y}(t))$ and $(p_{i,x}(t), p_{i,y}(t))$ are the position and momentum of the robot at time $t \in \mathbb{R}$, respectively. The robots possess a guidance law that steers them to their target positions $\bar{x}_i$ — albeit, without avoiding collisions nor maximizing a performance metric. The dynamics are pH with

$$V(\mathbf{x}(t)) = \sum_{i=1}^{M} V_i(x_i(t)), \quad V_i(x_i(t)) = (x_i(t) - \bar{x}_i)^{\mathsf{T}} U_i (x_i(t) - \bar{x}_i)\,,$$

where $U_i = \text{blkdiag}(\frac{1}{2m_i} I_2, \frac{k_i}{2} I_2)$ and $k_i \geq 0$ represents the virtual spring of the initial guidance law. We defer the detailed expression of the pH system to Appendix D. As the robots are not dynamically coupled, the graph $\mathcal{G}_d$ only has self-loops and thus $S_d = I_{12}$.

**Information constraints** We assume that robot $i$ can only receive information in real-time from robot $i-1$ and robot $i+1$[4] according to a circular communication graph $\mathcal{G}_c$ with adjacency matrix $S_c$ reported in (30) of Appendix D. Hence, we require output measurement and communication radii of $R_y = R_\xi = 1$. According to Theorem 3, we set $L_y = 1$ and $L_\xi = 0$ to satisfy (17). Accordingly, we impose $\mathbf{J}, \mathbf{R}_c, \mathbf{K} \in \text{blkSparse}(S_c)$ and select a fully separable $\Phi(\boldsymbol{\xi}(t)) = \sum_{i=1}^{M} \Phi_i(\xi_i(t), \theta_i(t))$. For this experiment, we choose the controller energy as in Galimberti et al. (2021a) given by

$$\Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}) = \sum_{i=1}^{M} \Phi_i(\xi_i(t), \theta_i(t)) = \sum_{i=1}^{M} \log\left(\cosh\left(W_i(t)\xi_i(t) + b_i(t)\right)\right)^{\mathsf{T}} \mathbf{1}\,. \tag{22}$$

**Cost function and collision avoidance** The loss function is composed of three main addends. The first addend, denoted as $\mathcal{L}_x$, is the standard control cost that penalizes the overall distance of robots from their targets and the control effort over a time horizon $T \in \mathbb{R}$, and is given by

$$\mathcal{L}_x = \int_0^T \left[(\mathbf{x}(t) - \bar{\mathbf{x}})^\top \mathbf{Q}(t) \left(\mathbf{x}(t) - \bar{\mathbf{x}}\right)^\top + \mathbf{u}^\top(t) \mathbf{R}(t)\, \mathbf{u}(t)\right] dt\,. \tag{23}$$

The second addend, denoted as $\mathcal{L}_{ca}$, strongly penalizes the event that any two robots $i$ and $j$ find themselves at a distance $d_{ij}(t) < D$, where $D \in \mathbb{R}$ is a pre-defined safety distance. Accordingly,

$$\mathcal{L}_{ca} = \int_0^T \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} L_{ca}^{ij}(t)\, dt\,, \tag{24}$$

where $L_{ca}^{ij}(t) = (d_{ij}(t) + \epsilon)^{-2}$ if $d_{ij}(t) \leq D$, and $L_{ca}^{ij}(t) = 0$ otherwise, and $\epsilon > 0$ is a small constant such that $L_{ca}^{i,j}(t) < \infty$ at all times. The third addend of the loss function is a regularization term $\mathcal{R}_w = \int_0^T \sum_{i=1}^{M} \|W_i(t + \Delta) - W_i(t)\|^2\, dt$ that encourages smooth weight variations in time, and $\Delta > 0$ is a small value that acts as a discretization step-size. Overall, the total loss function is expressed as $\mathcal{L} = \mathcal{L}_x + \alpha_{ca}\mathcal{L}_{ca} + \alpha_w \mathcal{R}_w$, where $(\alpha_{ca}, \alpha_w) > 0$ are hyperparameters.

4. Technically, $\mathbf{mod}(i-1, M)$ and $\mathbf{mod}(i+1, M)$, where $\mathbf{mod}(\cdot)$ indicates the modulo operation.

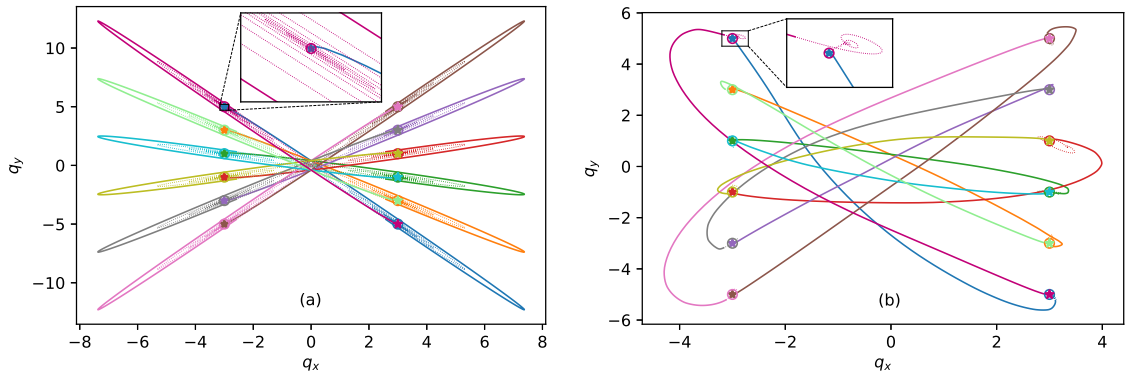


Figure 1: Trajectories of the robots in the $xy$-plane, (a) before training, and (b) after training a distributed H-DNN controller. Dotted lines indicate the trajectories for $t > T$, where $T$ is the training horizon. Target positions indicated with $\circ$. Initial positions indicated with $\star$.

Figure 1(a) shows robot trajectories before training; target positions are not reached within $T = 5$ seconds and multiple collisions occur around $(0, 0)$. Figure 1(b) shows trajectories after training a distributed H-DNN controller (13)-(14) with energy parametrized as per (22). The target positions $\circ$ are reached within the time horizon of $T = 5$ seconds and collisions are avoided while also optimizing the control cost $\mathcal{L}_x$. Videos associated with Figure 1 are available here. We note that distributed feedback policies favor robustness. Using the trained weights for perturbed initial states sampled from a radius $r = 0.5$ around $\mathbf{x}(0)$, collisions are avoided in $95\%$ of the runs.

**Comparison with distributed MLPs (Gama and Sojoudi, 2021)** We compare the closed-loop stability properties of H-DNN controllers Corollary 2 with a trained distributed MLP controller as per Gama and Sojoudi (2021).[5] We refer the reader to Appendix E for full details and results. Upon simulating the closed-loop system for a long horizon of $10T$, a trained MLP may destabilize the system for $t > T$. Instead, an H-DNN preserves stability as predicted by Corollary 2. For a fair comparison with distributed MLP controllers, we have also trained a *time-invariant* H-DNN with a comparable number of training parameters, and observed the same phenomenon after training.

**Non-vanishing gradients** We computed the norms of the terms $\frac{\partial \boldsymbol{\zeta}_j}{\partial \boldsymbol{\zeta}_i}$ for all $i, j$ satisfying $0 \leq i < j \leq N$ at every iteration, and verified that gradients never tend to vanish, despite a high number of neural ODE layers and dissipation in the dynamics. We refer the reader to Appendix F for details.

## 6. Conclusions

As we move towards highly nonlinear deep distributed control for safety-critical cyberphysical systems, dependability during exploration becomes a primary concern. We have proposed dependable and distributed H-DNN control policies that preserve closed-loop stability and non-vanishing gradients for arbitrarily large networks of nonlinear pH systems. Near-optimal performance can be achieved thanks to parametrizing deep nonlinear Hamiltonian energy functions for the controllers.

5. Without the structural assumptions needed for a scalable GNN implementation.

## Acknowledgments

We thank the anonymous reviewers for their precious insights regarding potential continuations and implications of this work. This research is supported by the Swiss NSF under the NCCR Automation (grant agreement 51NF40_180545).

## References

Shahbaz Abdulkhader, Hang Yin, Pietro Falco, and Danica Kragic. Learning deep energy shaping policies for stability-guaranteed manipulation. *IEEE Robotics and Automation Letters*, 2021.

Martin Angerer, Selma Musić, and Sandra Hirche. port-Hamiltonian based control for human-robot team interaction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2292–2299. IEEE, 2017.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128, 2016.

Felix Berkenkamp, Matteo Turchetta, Angela P Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in Neural Information Processing Systems 30*, 2:909–919, 2018.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *arXiv preprint arXiv:2108.06266*, 2021.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

Krzysztof Choromanski, Jared Quincy Davis, Valerii Likhosherstov, Xingyou Song, Jean-Jacques Slotine, Jacob Varley, Honglak Lee, Adrian Weller, and Vikas Sindhwani. An ode to an ODE. *arXiv preprint arXiv:2006.11421*, 2020.

Thai Duong and Nikolay Atanasov. Hamiltonian-based neural ODE networks on the SE (3) manifold for dynamics learning and control. *arXiv preprint arXiv:2106.12782*, 2021.

Jean-Michel Fahmi and Craig A. Woolsey. port-Hamiltonian flight control of a fixed-wing aircraft. *IEEE Transactions on Control Systems Technology*, pages 1–8, 2021. doi: 10.1109/TCST.2021.3059928.

Clara Lucía Galimberti, Luca Furieri, Liang Xu, and Giancarlo Ferrari-Trecate. Hamiltonian deep neural networks guaranteeing non-vanishing gradients by design. *arXiv preprint arXiv:2105.13205*, 2021a.

Clara Lucía Galimberti, Liang Xu, and Giancarlo Ferrari-Trecate. A unified framework for Hamiltonian deep neural networks. In *Learning for Dynamics and Control*, pages 275–286. PMLR, 2021b.

Fernando Gama and Somayeh Sojoudi. Graph neural networks for distributed linear-quadratic control. In *Learning for Dynamics and Control*, pages 111–124. PMLR, 2021.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

Yuqian Guo and Daizhan Cheng. Stabilization of time-varying Hamiltonian systems. *IEEE Transactions on Control Systems Technology*, 14(5):871–880, 2006.

Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled Cayley transform. In *International Conference on Machine Learning*, pages 1969–1978. PMLR, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Hassan K Khalil. Nonlinear systems third edition. *Patience Hall*, 115, 2002.

Arbaaz Khan, Ekaterina Tolstaya, Alejandro Ribeiro, and Vijay Kumar. Graph policy gradients for large scale robot control. In *Conference on robot learning*, pages 823–834. PMLR, 2020.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control (CDC)*, pages 6059–6066. IEEE, 2018.

Lukas Kölsch, Pol Jané Soneira, Felix Strehle, and Sören Hohmann. Optimal control of port-Hamiltonian systems: A continuous-time learning approach. *Automatica*, 130:109725, 2021.

Laurent Lessard and Sanjay Lall. Quadratic invariance is necessary and sufficient for convexity. In *IEEE American Control Conference (ACC)*, pages 5360–5362, 2011.

Abeje Y. Mersha, Raffaella Carloni, and Stefano Stramigioli. Port-based modeling and control of underactuated aerial vehicles. In *2011 IEEE International Conference on Robotics and Automation*, pages 14–19, 2011. doi: 10.1109/ICRA.2011.5980053.

LE Muñoz, Omar Santos, Pedro Castillo, and Isabelle Fantoni. Energy-based nonlinear control for a quadrotor rotorcraft. In *2013 American Control Conference*, pages 1177–1182. IEEE, 2013.

Romeo Ortega, Arjan Van Der Schaft, Fernando Castanos, and Alessandro Astolfi. Control by interconnection and standard passivity-based control of port-Hamiltonian systems. *IEEE Transactions on Automatic control*, 53(11):2527–2542, 2008.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

Patricia Pauli, Johannes Köhler, Julian Berberich, Anne Koch, and Frank Allgöwer. Offset-free setpoint tracking using neural network controllers. In *Learning for Dynamics and Control*, pages 992–1003. PMLR, 2021.

Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, pages 466–476. PMLR, 2018.

Michael Rotkowitz and Sanjay Lall. A characterization of convex problems in decentralized control. *IEEE Transactions on Automatic Control*, 51(2):274–286, 2006.

Olivier Sprangers, Robert Babuška, Subramanya P Nageshrao, and Gabriel AD Lopes. Reinforcement learning for port-Hamiltonian systems. *IEEE Transactions on Cybernetics*, 45(5):1017–1027, 2014.

Felix Strehle, Pulkit Nahata, Albertus Johannes Malan, Sören Hohmann, and Giancarlo Ferrari-Trecate. A unified passivity-based framework for control of modular islanded AC microgrids. *IEEE Transactions on Control System Technology, to appear. arXiv preprint arXiv:2104.02637*, 2021.

Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In *Conference on robot learning*, pages 671–682. PMLR, 2020.

Arjan J Van der Schaft. *L2-gain and passivity techniques in nonlinear control*, volume 2. Springer, 2000.

Arjan J van der Schaft. Port-Hamiltonian systems: network modeling and control of nonlinear physical systems. In *Advanced dynamics and control of structures and machines*, pages 127–167. Springer, 2004.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, pages 3570–3578. PMLR, 2017.

Hans S Witsenhausen. A counterexample in stochastic optimum control. *SIAM Journal on Control*, 6(1):131–147, 1968.

Fengjun Yang and Nikolai Matni. Communication topology co-design in graph recurrent neural network based distributed control. *arXiv preprint arXiv:2104.13868*, 2021.

## Appendix A. Proof of Proposition 1

When $\mathbf{u}(t) = 0$, the networked system can be rewritten as

$$\dot{\mathbf{x}}(t) = (\mathbf{\Omega} + \mathbf{FG} - \mathbf{R}) \frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}(t)} \,. \tag{25}$$

If $\mathbf{FG}$ is skew-symmetric, we have

$$\begin{aligned} \dot{V}(\mathbf{x}(t)) &= \frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}} \dot{\mathbf{x}}(t) = -\frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}}^{\mathsf{T}} \mathbf{R} \frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}} \\ &= -\sum_{i=1}^{M} \frac{\partial V_i(x_i(t))}{\partial x_i}^{\mathsf{T}} R_i \frac{\partial V_i(x_i(t))}{\partial x_i} \leq 0 \,. \end{aligned} \tag{26}$$

Since $V_i(\cdot)$ is radially unbounded for every $i = 1, \ldots, M$, then $V(\cdot)$ is radially unbounded. Hence, all level sets of $V(\cdot)$ are bounded, and since $V(\mathbf{x}(0)) < \infty$ and $\dot{V}(\mathbf{x}(t)) \leq 0$ for every $t$ all system trajectories are bounded. By Krasovski-LaSalle's Invariance Principle (Khalil, 2002) we conclude that, for any initial state $\mathbf{x}(0)$, the state $\mathbf{x}(t)$ converges to the largest invariant set contained in

$$\begin{aligned} E &= \left\{ \overline{\mathbf{x}} | \ \frac{\partial V(\overline{\mathbf{x}})}{\partial \mathbf{x}}^{\mathsf{T}} \mathbf{R} \frac{\partial V(\overline{\mathbf{x}})}{\partial \mathbf{x}} = 0 \right\} \\ &= \left\{ \overline{\mathbf{x}} | \ \frac{\partial V_i(\overline{x}_i)}{\partial x_i}^{\mathsf{T}} R_i \frac{\partial V_i(\overline{x}_i)}{\partial x_i} = 0, \ \ \forall i = 1, \ldots, M \right\} \\ &= E_1 \times E_2 \times \ldots \times E_M \,, \end{aligned}$$

where, for every $i = 1, \ldots, M$, it holds that $E_i = \left\{ \overline{x}_i | \ \frac{\partial V_i(\overline{x}_i)}{\partial x_i} \in \mathrm{Ker}\,(R_i) \right\}$.

## Appendix B. Proof of Theorem 3

Let $J_{ij}$, $R_{c,ij}$ and $K_{ij}$ denote the blocks located at position $(i, j)$ of $\mathbf{J}$, $\mathbf{R}_c$ and $\mathbf{K}$, respectively. For any $i \in \{1, \ldots, M\}$, the local controller equation can be written as

$$\begin{aligned} \dot{\xi}_i(t) &= \sum_{j \in \mathcal{N}_i^{L_y}} \left[ (J_{ij} - R_{c,ij}) \frac{\partial \left( \sum_{l \in \mathcal{N}_j^{L_\xi}} \Phi_l \left( \boldsymbol{\xi}_l^{L_\xi}(t), \theta_l(t) \right) \right)}{\partial \xi_j} + K_{ij} y_j(t) \right] , \\ u_i(t) &= - \sum_{j \in \mathcal{N}_i^{L_y}} \left[ (K_{ji})^{\mathsf{T}} \frac{\partial \left( \sum_{l \in \mathcal{N}_j^{L_\xi}} \Phi_l \left( \boldsymbol{\xi}_l^{L_\xi}(t), \theta_l(t) \right) \right)}{\partial \xi_j} \right] . \end{aligned}$$

By inspection, we conclude that $\dot{\xi}_i(t)$ is computed as a function of the internal controller variables $\xi_l(t)$ of at most its $(L_y + 2L_\xi)$-hops neighbors and the output measurements of at most its $L_y$-hops neighbors. Further, the local control input $u_i(t)$ is computed as a function of the internal controller variables $\xi_l(t)$ of at most its $(L_y + 2L_\xi)$-hops neighbors. The condition (17) follows.

## Appendix C. Proof of Corollary 4

Let $Z \triangleq \frac{\partial \boldsymbol{\zeta}(\tilde{t})}{\partial \boldsymbol{\zeta}(\tilde{t}-t)}$ for brevity and $g(\boldsymbol{\zeta}(t), \boldsymbol{\theta}(t)) = \boldsymbol{\Psi} \frac{\partial P(\boldsymbol{\zeta}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\zeta}}$. By adapting Lemma 1 of Galimberti et al. (2021a), we have that

$$\dot{Z} = \frac{\partial^{\mathsf{T}} g(\boldsymbol{\zeta}(\tilde{t}-t), \boldsymbol{\theta}(\tilde{t}-t))}{\partial \boldsymbol{\zeta}} Z \,. \tag{27}$$

By (20), (27) and $\boldsymbol{\Psi} = -\boldsymbol{\Psi}^{\mathsf{T}}$ we obtain

$$\begin{aligned}
&\frac{d}{dt}\left(Z^{\top}\boldsymbol{\Psi} Z\right) = \dot{Z}^{\top}\boldsymbol{\Psi} Z + Z^{\top}\boldsymbol{\Psi}\dot{Z} \\
&= Z^{\mathsf{T}} \frac{\partial g(\boldsymbol{\zeta}(\tilde{t}-t), \boldsymbol{\theta}(\tilde{t}-t))}{\partial \boldsymbol{\zeta}} \boldsymbol{\Psi} Z + Z^{\mathsf{T}} \boldsymbol{\Psi} \frac{\partial^{\mathsf{T}} g(\boldsymbol{\zeta}(\tilde{t}-t), \boldsymbol{\theta}(\tilde{t}-t))}{\partial \boldsymbol{\zeta}} Z \\
&= Z^{\mathsf{T}} \left( \boldsymbol{\Psi} \frac{\partial^2 P(\boldsymbol{\zeta}(\tilde{t}-t), \boldsymbol{\theta}(\tilde{t}-t))}{\partial \boldsymbol{\zeta}^2} \boldsymbol{\Psi} + \boldsymbol{\Psi} \frac{\partial^2 P(\boldsymbol{\zeta}(\tilde{t}-t), \boldsymbol{\theta}(\tilde{t}-t))}{\partial \boldsymbol{\zeta}^2} \boldsymbol{\Psi}^{\mathsf{T}} \right) Z = 0 \,.
\end{aligned}$$

Since $\frac{\partial \boldsymbol{\zeta}(\tilde{t})}{\partial \boldsymbol{\zeta}(\tilde{t})} = I$ and the quantity $Z^{\mathsf{T}}\boldsymbol{\Psi} Z$ is constant over time, we deduce that

$$\|\boldsymbol{\Psi}\| = \left\| Z^{\mathsf{T}} \boldsymbol{\Psi} Z \right\| \leq \|Z\|^2 \|\boldsymbol{\Psi}\| \,,$$

and hence $\|Z\| \geq 1$ for every $t$ and $\tilde{t}$ such that $t \leq \tilde{t} \leq T$ for any $T \in \mathbb{R}$.

## Appendix D. Implementation

We consider a fleet of $M = 12$ mobile robots that need to achieve a pre-specified formation described by $(\bar{q}_{i,x}, \bar{q}_{i,y})$ for each agent $i$ and with zero velocity. Each vehicle $i$ endowed with a guidance law is described by

$$\begin{bmatrix} \dot{p}_{i,x} \\ \dot{p}_{i,y} \\ \dot{q}_{i,x} \\ \dot{q}_{i,y} \end{bmatrix} = \left( \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} - \begin{bmatrix} b_i & 0 & 0 & 0 \\ 0 & b_i & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} \frac{\partial V_i}{\partial p_{i,x}} \\ \frac{\partial V_i}{\partial p_{i,y}} \\ \frac{\partial V_i}{\partial q_{i,x}} \\ \frac{\partial V_i}{\partial q_{i,y}} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} u_i \tag{28}$$

where

$$V_i(t) = \frac{1}{2m_i}\left( (p_{i,x}(t))^2 + (p_{i,y}(t))^2 \right) + \frac{1}{2} k_i \left( (q_{i,x}(t) - \bar{q}_{i,x})^2 + (q_{i,y}(t) - \bar{q}_{i,y})^2 \right) , \tag{29}$$

for $i = 1, 2, \ldots, M$ with $m_i = k_i = 1$ and $b_i = 0.2$. The inputs $u_i = (F_{i,x}, F_{i,y})$ are forces in the $x$ and $y$ directions. The initial conditions of the systems are fixed and indicated in Figure 1 with a "$\star$" symbol.

The adjacency matrix $S_c$ of the communication graph $\mathcal{G}_c$ is defined as

$$S_c = \begin{bmatrix} 1 & 1 & 0 & 0 & \ldots & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \ldots & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & \ldots & 0 & 1 & 1 \end{bmatrix} \in \mathbf{R}^{12 \times 12} \,. \tag{30}$$

The distributed H-DNN controller is given by (13)-(14) where $\mathbf{R}_c = \text{blkdiag}(R_{c,1}, R_{c,2}, \ldots, R_{c,12})$, $R_{c,i} = 12\left[\begin{smallmatrix} I_2 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ for $i = 1 \ldots, 12$. We let $\mathbf{J}$ satisfy $\mathbf{J} = -\mathbf{J}^\top$ and $\mathbf{K}$ be trainable matrix parameters that lie in $\text{blkSparse}(S_c)$.

The controller energy is completely separable to comply with $L_\xi = 0$, and defined as $\Phi(\boldsymbol{\xi}(t), \boldsymbol{\theta}(t)) = \sum_{i=1}^{12} \Phi_i(\xi_i, \theta_i)$ with $\xi_i \in \mathbb{R}^4$. We define the local energies as

$$\Phi_i(\xi_i, \theta_i) = \tilde{\sigma}(W_i(t)\xi_i(t) + b_i(t))^\mathsf{T}\mathbf{1}\,, \quad \forall i = 1, \ldots, 12\,, \tag{31}$$

where $\theta_i = (W_i, b_i)$ and $\tilde{\sigma}(\cdot) = \log(\cosh(\cdot))$ is applied element-wise.

We implement the training of (19)-(20) as a Neural ODE endowed with forward Euler discretization. Setting $N = 100$ step times, we obtain a time step $h = T/N = 0.05$ and 100 layers for the neural network. The trainable parameters $\mathbf{W}(t)$ and $\mathbf{b}(t)$ are modelled as piece-wise constant functions in each interval of time $[t_k, t_k + h]$ for $t_k = k \cdot h$ and $k = 0, 1, \ldots, N-1$. During training, we fixed $\xi_i(0) = (3, 0, 0, 0)$ for all robots $i$ and simulated trajectories for the same initial conditions $\boldsymbol{\zeta}(0)$ for all iterations of gradient descent.

To validate the closed-loop stability properties of our H-DNN controllers in comparison with standard MLPs from Gama and Sojoudi (2021), we need to simulate the system for $t > T$. For long-horizon simulations, we freeze $\boldsymbol{\theta}(t) = \boldsymbol{\theta}(T)$ for all times $t > T$, and in order to faithfully simulate the continuous-time closed-loop we use the Runge-Kutta 5 integration method. Specifically, all the losses reported in Table 1 are calculated using Runge-Kutta 5 integration.

We use gradient descent with Adam for 300 epochs, in order to minimize the loss function

$$\mathcal{L} + \alpha_{ca}\mathcal{L}_{ca} + \alpha_w \mathcal{R}_w\,, \tag{32}$$

with hyperparameters $\alpha_{ca} = 100$ and $\alpha_w = 125 - \alpha_{ca}$. Moreover, for the collision avoidance term, we set a minimum distance $D = 0.5$, and for the control loss (23) we set $R(t) = 0.5\, I$ and $Q(t) = \gamma^{T-t} I$ with $\gamma = 0.95$ during training and $\gamma = 1$ for testing. Here, $\gamma$ acts as a discount factor that assigns less weight to late-horizon costs.

## Appendix E. MLP controller

We have compared our distributed H-DNN controller with an MLP distributed controller as per Gama and Sojoudi (2021).[6] To keep the comparison fair, we have trained a *time-invariant* H-DNN — rather than a time-varying one — so that the MLP and the H-DNN possess a comparable number of training parameters.

**Time-invariant (TI) H-DNN implementation details** In the TI H-DNNs, we force parameters to be constant across layers, i.e., $\mathbf{W}(t) = \mathbf{W}$ and $\mathbf{b}(t) = \mathbf{b}$ for all $t \in \mathbb{R}$. Moreover, we select the energy function $\Phi_i(\cdot)$ as a 5-layered NN as per

$$\begin{aligned} &\omega_i^0 = \xi_i(t)\,, \\ &\omega_i^{k+1} = \log(\cosh(W_i^k \omega_i^k + b_i^k)) \quad \text{for } k = 0, \ldots, 4\,, \\ &\Phi_i(\xi_i, \theta_i) = (\omega_i^5)^\mathsf{T}\mathbf{1}\,, \end{aligned}$$

where $\mathbf{1}$ denotes the vector of ones of appropriate dimensions.

6. Without the structural assumptions needed for a scalable GNN implementation.

**MLP implementation details** The implemented distributed MLP of Table 1 is a 3-layered NN with weights $\mathbf{W}^j$ and bias $\mathbf{b}^j$ for $j = 0, \dots, 2$, and $\tanh(\cdot)$ as activation function. The layer dimensions at each layer are given by

$$2 \xrightarrow{W_i^0} 8 \xrightarrow{W_i^1} 8 \xrightarrow{W_i^2} 2\,. \tag{33}$$

In order to comply with the communication graph given by $S_c$, we set $\mathbf{W}^0 \in \text{blkSparse}(S_c)$ and $\mathbf{W}^j, \in \text{blkSparse}(I_{12})$ for $j = 1, 2$ as suggested in Gama and Sojoudi (2021).

The results are reported in Table 1 and Figures 2 and 3. While an MLP controller might achieve a better performance within the training horizon $T$ due to fewer structural assumptions, closed-loop stability cannot be guaranteed even after training. Note that this comparison holds both for the previous setting as well as for a much simpler task where we only minimize $\mathcal{L}_x$, without the collision avoidance task.[7]

In confirmation of the closed-loop stability result of Corollary 2, Figure 1(b) shows that using an H-DNN controller the robots asymptotically reach their target locations for $t > T$. Furthermore, Table 1 shows that despite simulating for a longer horizon of $10T$ the cumulative loss only increases by $3.7\%$. Instead, a trained MLP controller might introduce persistently oscillating trajectories with large amplitude for $t > T$ — as reported in Figure 2 and Figure 3. Correspondingly, the cumulative loss for a horizon of $10T$ may increase by more than 30 times as we indicate in **red color** in Table 1.

We last note that, to keep the comparison fair, we have trained the TI H-DNN controller and the MLP controller using the same optimizer and hyperparameters. To improve the performance of the MLP controller, we have trained it for twice as many epochs as the (TI) H-DNN controller, i.e., 600 epochs instead of 300 epochs.

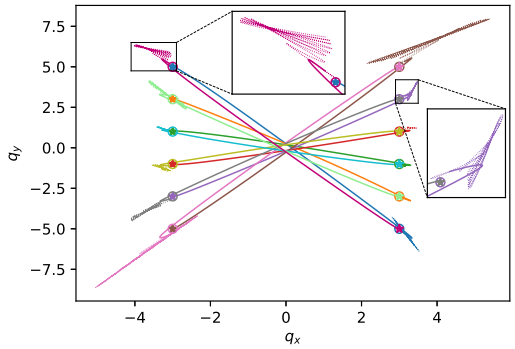


Figure 2: Trajectories of the robots in the *xy*-plane when using a distributed MLP controller without c.a. Solid line: training time horizon. Dotted line: extended time horizon.

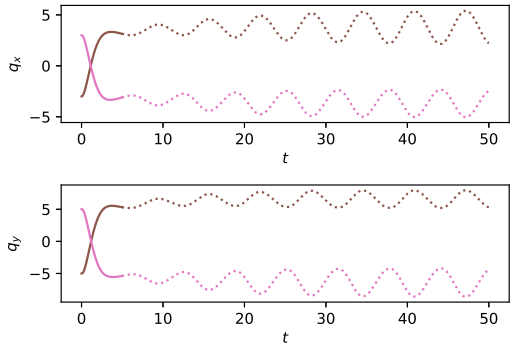


Figure 3: Trajectories of the *pink* and *brown* robots when using a distributed MLP controller without c.a. Solid line: training time horizon. Dotted line: extended time horizon.

## Appendix F. Gradient norms

To validate the result of Corollary 4, we computed the norms of the terms $\frac{\partial \boldsymbol{\zeta}_j}{\partial \boldsymbol{\zeta}_i}$ for all $i, j$ satisfying $0 \leq i < j \leq N$ at every iteration. We verified that gradients never tend to vanish, despite a high number of neural ODE layers and dissipation in the dynamics. Figure 4 and Figure 5 show

7. That is, when we set $\alpha_{ca} = 0$.

Table 1: Cumulative cost $\mathcal{L}_x$ when comparing a distributed H-DNN with an MLP controller, with and without collision avoidance (c.a.). Training performed for a time horizon of $T = 5$.

| | Simulation time | H-DNN controller | H-DNN TI controller | MLP controller |
|---|---|---|---|---|
| with c.a. | $T$ | 34358 | 38355 | 37308 |
| | $10\,T$ | 34618 | 39232 | **1205319** |
| without c.a. | $T$ | 31611 | 31446 | 31433 |
| | $10\,T$ | 32769 | 31494 | **51133** |
| # of trainable parameters | | 24480 | 1680 | 1944 |

the sensitivities for the case $j = N$ and $j = N/2$ respectively. As the sensitivity norms do not drop below the level $\approx 10$ despite a large number of neural-network layers, both plots confirm our results of Corollary 4 even if some dissipation is present in the system dynamics. Note also that Figure 4 highlights that the most sensitive gradients are with respect to $\zeta_\ell$ for $\ell \in [30, 40]$ which coincides with the time interval where the distances between agents decreased. Thus, it is expected that these loss terms are more sensitive to variations. In the case of Figure 5, we can appreciate that all gradients are changing during training. This highlights that sensitivity matrices are modified during the learning of optimal trajectories.

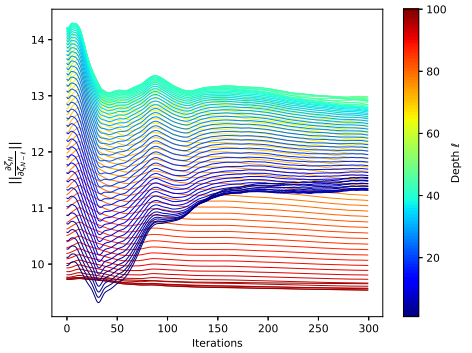


Figure 4: Norm of the BSMs $\frac{\partial \boldsymbol{\zeta}_N}{\partial \boldsymbol{\zeta}_{N-j}}$ for $j = 1, \ldots, 100$.

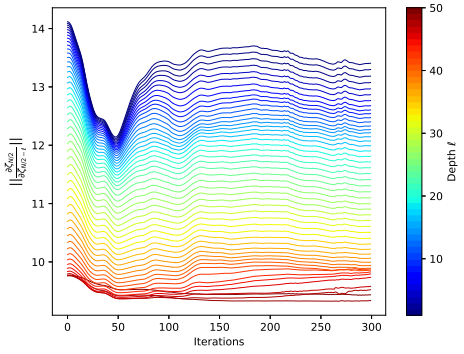


Figure 5: Norm of the BSMs $\frac{\partial \boldsymbol{\zeta}_{N/2}}{\partial \boldsymbol{\zeta}_{N/2-j}}$ for $j = 1, \ldots, 50$.