Appendix A. Proof of Proposition 1

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

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

If $\mathbf{F}\mathbf{G}$ 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

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

$$u_i(t) = -\sum_{j\in\mathcal{N}_i^{L_y}}\left[(K_{ji})^{\mathsf{T}}\frac{\partial\left(\sum_{l\in\mathcal{N}_j^{L_\xi}}\Phi_l\left(\boldsymbol{\xi}_l^{L_\xi}(t), \theta_l(t)\right)\right)}{\partial \xi_j}\right]\,.$$

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 Theorem 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 $\mathbf{R}_c = \mathrm{blkdiag}(R_{c,1}, R_{c,2}, \ldots, R_{c,12})$, $R_{c,i} = 12\left[\begin{smallmatrix} I_2 & 0 \\ 0 & 0 \end{smallmatrix}\right]$ 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 $\mathrm{blkSparse}(S_c)$.

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

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

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, γ 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

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

[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.

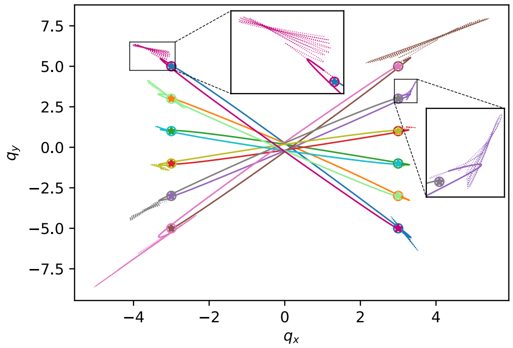

Figure 2: Trajectories of the robots in the *xy*-plane when using a distributed MLP controller without c.a. Solid line: training time horizon. Dotted line: extended time horizon.

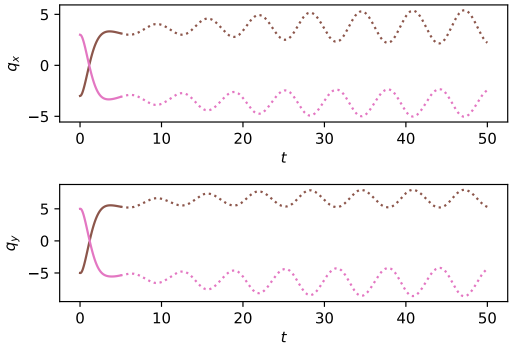

Figure 3: Trajectories of the *pink* and *brown* robots when using a distributed MLP controller without c.a. Solid line: training time horizon. Dotted line: extended time horizon.

Appendix F. Gradient norms

To validate the result of Theorem 4, we computed the norms of the terms $\frac{\partial \zeta_j}{\partial \zeta_i}$ 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 ≈ 10 despite a large number of neural-network layers, both plots confirm our results of Theorem 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 ζ_ℓ 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.

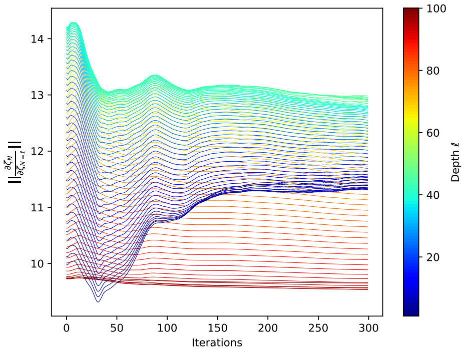

Figure 4: Norm of the BSMs $\frac{\partial \zeta_N}{\partial \zeta_{N-j}}$ for $j = 1, \ldots, 100$

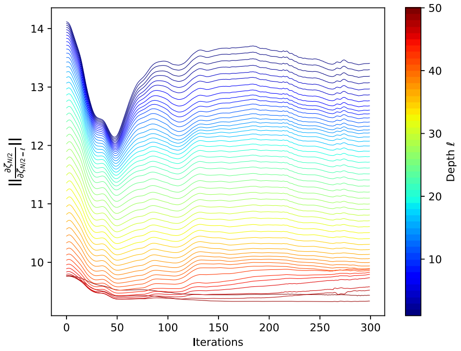

Figure 5: Norm of the BSMs $\frac{\partial \zeta_{N/2}}{\partial \zeta_{N/2-j}}$ for $j = 1, \ldots, 50$