Neural System Level Synthesis: Learning over All Stabilizing Policies for Nonlinear Systems

Implementation details

In this document, we introduce the implementation details concerning the numerical experiments of Furieri et al. (2022a). *Notation:* for vectors $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, $(a, b) = \left[a^\top, b^\top\right]^\top$ is a vector in $\mathbb{R}^{n+m}$.

We consider point-mass vehicles with state (p_t, q_t) where $p_t \in \mathbb{R}^2$ represents the position and $q_t \in \mathbb{R}^2$ the velocity. The agents are subject to nonlinear drag forces (e.g., air or water resistance). The discrete-time model for each vehicle of mass $m = 1\,\mathrm{kg}$ is

$$\begin{bmatrix} p_t \\ q_t \end{bmatrix} = \begin{bmatrix} p_{t-1} \\ q_{t-1} \end{bmatrix} + T_s \begin{bmatrix} q_{t-1} \\ m^{-1}\left(-C(q_{t-1})q_{t-1} + F_{t-1}\right) \end{bmatrix}, \tag{1}$$

where $F_t \in \mathbb{R}^2$ denotes the force control input, $T_s = 0.05\,\mathrm{s}$ is the sampling time and $C : \mathbb{R}^2 \to \mathbb{R}$ is a positive *drag function*. We remark that the disturbance sequence of the Neur-SLS problem Furieri et al. (2022a) is given by $\mathbf{w} = ((p_0, q_0, 0, 0), (0, 0, 0, 0), \dots) \in \ell_2^4$.

The vehicles need to achieve a target position $\overline{p} \in \mathbb{R}^2$ with zero velocity, i.e., $\overline{q} = 0_2$, in a finite time of $5\,\mathrm{s}$, resulting in $T = 100$ time-steps. For this setting, we consider a base controller $u_t = K'(\overline{p} - p_t)$ with $K' = \mathrm{diag}(k_1', k_2')$ and $k_1' = k_2' = 1\,\frac{\mathrm{N}}{\mathrm{m}}$.

We model a set of $N \in \mathbb{N}$ vehicles as per (1) by defining an overall state $x_t = \left(p_t^1, q_t^1, p_t^2, q_t^2, \dots, p_t^N, q_t^N\right) \in \mathbb{R}^{4N}$ and input $u_t = \left(F_t^1, F_t^2, \dots, F_t^N\right) \in \mathbb{R}^{2N}$. We consider two scenarios: `mountains` and `swapping`.[1] Animations are available in our Github repository.[2]

For each scenario, we train Neur-SLS control policies to optimize the performance given by

$$\sum_{t=0}^{T} l(x_t^s, u_t^s) = \sum_{t=0}^{T} l_{traj}(x_t, u_t) + l_{ca}(x_t) + l_{obs}(x_t) \tag{2}$$

where

$$l_{traj}(x_t, u_t) = \sum_{i=0}^{N} \left(p_t^i, q_t^i\right)^\mathsf{T} Q \left(p_t^i, q_t^i\right) + \alpha_u \left(F_t^i\right)^\mathsf{T} \left(F_t^i\right), \tag{3}$$

$$l_{ca}(x_t) = \begin{cases} \alpha_{ca} \sum_{i=0}^{N} \sum_{j,\, i \neq j} (d_{ij}(t) + \epsilon)^{-2} & \text{if} \quad d_{ij}(t) \leq D\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{4}$$

where $Q \succeq 0$, d_{ij} represents the distance between agent i and j and ϵ is a fixed positive small constant such that the loss is well defined for all positive distances. The last addend of (2), represents

1. The `mountains` and `swapping` benchmarks are motivated by the examples in Onken et al. (2021); Furieri et al. (2022b).
2. `https://github.com/DecodEPFL/neurSLS.git`

the cost due to obstacles in the environment. We define the obstacles using a Gaussian density function

$$\eta(p; \mu, \Sigma) = \frac{1}{2\pi\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(p-\mu)^\top \Sigma^{-1} (p-\mu)\right) \tag{5}$$

with mean $\mu \in \mathbb{R}^2$ and covariance $\Sigma \in \mathbb{R}^{2\times 2}$. Then, the term $l_{obs}(x_t^i)$ is given by

$$l_{obs}(x_t) = \alpha_{obst} \sum_{i=0}^{N} \left(\eta\left(p_t^i; \begin{bmatrix} 2.5 \\ 0 \end{bmatrix}, 0.2\, I\right) + \eta\left(p_t^i; \begin{bmatrix} -2.5 \\ 0 \end{bmatrix}, 0.2\, I\right)\right. \tag{6}$$

$$\left. + \eta\left(p_t^i; \begin{bmatrix} 1.5 \\ 0 \end{bmatrix}, 0.2\, I\right) + \eta\left(p_t^i; \begin{bmatrix} -1.5 \\ 0 \end{bmatrix}, 0.2\, I\right)\right). \tag{7}$$

We use stochastic gradient descent with Adam in order to minimize the loss function. For the `mountains` scenario, we set the hyperparameters $Q = \mathrm{diag}(1, 1, 1, 1)$, $\alpha_u = 0.1$, $\alpha_{ca} = 100$ and $\alpha_{obst} = 5000$, and train for 500 epochs with a learning rate of 0.001. For the `swapping` problem, we train during 1500 epochs with learning rate equal to 0.002 and set the hyperparameters to be $Q = \mathrm{diag}(1, 1, 1, 1)$, $\alpha_u = 0.1$ and $\alpha_{ca} = 1000$. Since there are no fixed obstacles in the environment, we set $\alpha_{obst} = 0$.

MOUNTAINS PROBLEM (2 ROBOTS)

The scenario `mountains` involves two agents whose goal is to coordinately pass through a narrow valley. The system consists of 2 robots of radius $0.5\,\mathrm{m}$ and we consider a drag function given by

$$C(q)q = b_1 q + b_2 |q| q\,, \tag{8}$$

with $b_1 = 1\,\frac{\mathrm{N\,s}}{\mathrm{m}}$ and $b_2 = 0.1\,\frac{\mathrm{N\,s}}{\mathrm{m}}$.

The REN is a deep neural network with depth $r = 32$ layers ($v \in \mathbb{R}^r$). Its internal state ξ is of dimension $q = 32$. We use $\tanh(\cdot)$ as the activation function.

The initial positions of the robots are sampled from a Normal distribution centered at $(\pm 2, -2)$, with covariance $\mathrm{diag}(\sigma^2, \sigma^2)$. We set $\sigma = 0.2$ for the first 300 epochs and then increased it to $\sigma = 0.5$. At each epoch we simulate five trajectories over which we calculate the corresponding loss.

SWAPPING PROBLEM (12 ROBOTS)

The scenario `swapping` considers twelve agents switching their positions, while avoiding all collisions. The system consists of 12 robots of radius $0.25\,\mathrm{m}$ and the considered drag function is given by

$$C(q)q = bq\,, \tag{9}$$

with $b = 1\,\frac{\mathrm{N\,s}}{\mathrm{m}}$.

The REN is a deep neural network with depth $r = 24$ layers ($v \in \mathbb{R}^r$). Its internal state ξ is of dimension $q = 96$. We use $\tanh(\cdot)$ as the activation function.

References

Luca Furieri, Clara Lucía Galimberti, and Giancarlo Ferrari-Trecate. Neural system level synthesis: Learning over all stabilizing policies for nonlinear systems. *under review*, 2022a.

Luca Furieri, Clara Lucía Galimberti, Muhammad Zakwan, and Giancarlo Ferrari-Trecate. Distributed neural network control with dependability guarantees: a compositional port-Hamiltonian approach. *PMLR, to appear*, 2022b.

Derek Onken, Levon Nurbekyan, Xingjian Li, Samy Wu Fung, Stanley Osher, and Lars Ruthotto. A neural network approach applied to multi-agent optimal control. In *IEEE European Control Conference (ECC)*, pages 1036–1041, 2021.