# Closing the loop between nonlinear distributed control and learning-based optimization


## Dr. Luca Furieri


Swiss National Science Foundation

EPFL

ETH zürich

- SNSF Ambizione Fellow at EPF Lausanne (Jan. 2023 to present)
  - Principal investigator of «*Reliable Machine Learning for Distributed Control*»
- Postdoc at EPF Lausanne (Nov. 2020 to Dec. 2022)
  - Working with Prof. Giancarlo Ferrari Trecate
- PhD at ETH Zurich (Nov. 2016 to Sep 2020)
  - Supervised by Prof. Maryam Kamgarpour

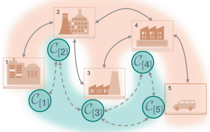
# Optimal control for large-scale dynamical systems

## Target challenges

# Optimal control for large-scale dynamical systems

## Target challenges

### 1) Optimality in distributed tasks

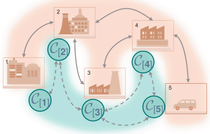



- *Myopic,* local improvement→ global effect

# Optimal control for large-scale dynamical systems

## Target challenges

### 1) Optimality in distributed tasks




- *Myopic,* local improvement→ global effect

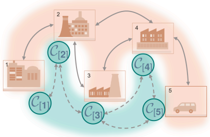
### 2) Uncertain system models

- *Noisy* data → optimal control policy
- Tension: performance VS guarantees

# Optimal control for large-scale dynamical systems

## Target challenges

### 1) Optimality in distributed tasks

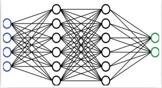


- *Myopic,* local improvement→ global effect

### 2) Uncertain system models

- *Noisy* data → optimal control policy
- Tension: performance VS guarantees

### 3) Guarantees of learning-based control



- Neural network policies, nonlinear objectives
- Closed-loop stability? Safety?

# Selected Contributions: Part 1

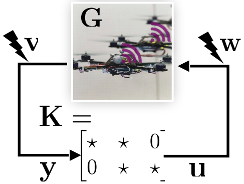
**Optimal Distributed Control (ODC) for Linear Systems**

[1] "*Sparsity invariance for convex design of distributed controllers*", Furieri, Zheng, Papachristodoulou, Kamgarpour, TCNS, 2020

[2] "*Learning the globally optimal distributed LQ regulator*", Furieri, Zheng, Kamgarpour, L4DC, 2020

# Beyond long-standing limitations of linear ODC[1]

[1] *"Sparsity Invariance for Convex Design of Distributed Controllers"*, Furieri, Zheng, Papachristodoulou, Kamgarpour, [TCNS20]*
*IEEE Transactions on Control of Network Systems Best Paper Award, 2022



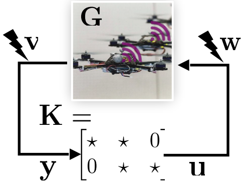

$$= \|(\mathbf{w},\mathbf{v}) \underset{\mathbf{G},\mathbf{K}}{\rightarrow} (\mathbf{u},\mathbf{y})\|_{\mathcal{H}_2/\mathcal{H}_\infty}$$

$$\min_{\mathbf{K}\text{ stabilizing}} \quad \text{cost}(\mathbf{G},\mathbf{K})$$

$$\text{subject to} \quad \mathbf{K} = \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$$

# Beyond long-standing limitations of linear ODC[1]

[1] *"Sparsity Invariance for Convex Design of Distributed Controllers"*, Furieri, Zheng, Papachristodoulou, Kamgarpour, [TCNS20]*
*IEEE Transactions on Control of Network Systems Best Paper Award, 2022




$$= ||(\mathbf{w},\mathbf{v}) \xrightarrow[\mathbf{G},\mathbf{K}]{} (\mathbf{u},\mathbf{y})||_{\mathcal{H}_2/\mathcal{H}_\infty}$$

$$\min_{\mathbf{K}\ \text{stabilizing}} \quad \text{cost}(\mathbf{G},\mathbf{K})$$

$$\text{subject to} \quad \mathbf{K} = \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$$

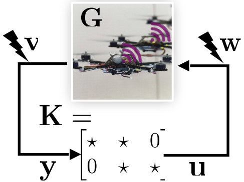
**Limitation:** convex reformulation available only if *Quadratic Invariance* (QI) holds [Rotkowitz et al., 2006]

"*design sparse closed-loop maps*"

"*sparse controller*" $\Longleftrightarrow$ "*sparse closed-loop maps*"

## Beyond long-standing limitations of linear ODC[1]

[1] *"Sparsity Invariance for Convex Design of Distributed Controllers"*, Furieri, Zheng, Papachristodoulou, Kamgarpour, [TCNS20]*
*IEEE Transactions on Control of Network Systems Best Paper Award, 2022




$$\mathbf{K} = \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$$

$$\min_{\mathbf{K} \text{ stabilizing}} \ \text{cost}(\mathbf{G}, \mathbf{K}) = ||(\mathbf{w}, \mathbf{v}) \underset{\mathbf{G},\mathbf{K}}{\rightarrow} (\mathbf{u}, \mathbf{y})||_{\mathcal{H}_2/\mathcal{H}_\infty}$$

$$\text{subject to} \quad \mathbf{K} = \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$$

**Limitation:** convex reformulation available only if *Quadratic Invariance* (QI) holds [Rotkowitz et al., 2006]

"*design sparse closed-loop maps*"

"*sparse controller*" $\Longleftrightarrow$ "*sparse closed-loop maps*"

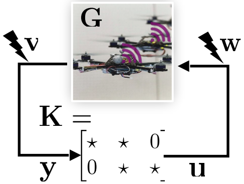
### My Contribution: near-optimal convex restrictions

1. QI as a special case of **Sparsity Invariance (SI)** ← Plant-independent
2. **Globally optimal and sparse** $\mathbf{K}$ for some non-QI cases
3. Near-optimality guarantees **for arbitrary sparsity patterns**

$\Longrightarrow$ **QI no-longer a limitation for sparse controller design**

## Beyond long-standing limitations of linear ODC[1]

[1] *“Sparsity Invariance for Convex Design of Distributed Controllers”*, Furieri, Zheng, Papachristodoulou, Kamgarpour, [TCNS20]*
*IEEE Transactions on Control of Network Systems Best Paper Award, 2022




$$\min_{\mathbf{K} \text{ stabilizing}} \quad \text{cost}(\mathbf{G}, \mathbf{K}) \qquad = \|(\mathbf{w},\mathbf{v}) \underset{\mathbf{G},\mathbf{K}}{\rightarrow} (\mathbf{u},\mathbf{y})\|_{\mathcal{H}_2/\mathcal{H}_\infty}$$

$$\text{subject to} \quad \mathbf{K} = \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$$

**Limitation:** convex reformulation available only if *Quadratic Invariance* (QI) holds [Rotkowitz et al., 2006]

*“des ... maps”*  *“sparse controller”* $\Longleftrightarrow$ *“sparse closed-loop maps”*

**Next question: unknown system model?**

### My Contribution: near-optimal convex restrictions

1. QI as a special case of **Sparsity Invariance (SI)** ← Plant-independent
2. **Globally optimal and sparse** $\mathbf{K}$ for some non-QI cases
3. Near-optimality guarantees **for arbitrary sparsity patterns**

$\Longrightarrow$ **QI no-longer a limitation for sparse controller design**

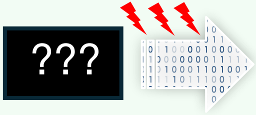
# Learning-based control philosophies

Model-based

Model-free
(RL-like)

# Learning-based control philosophies

Model-based




Noisy trajectories

**System Identification**

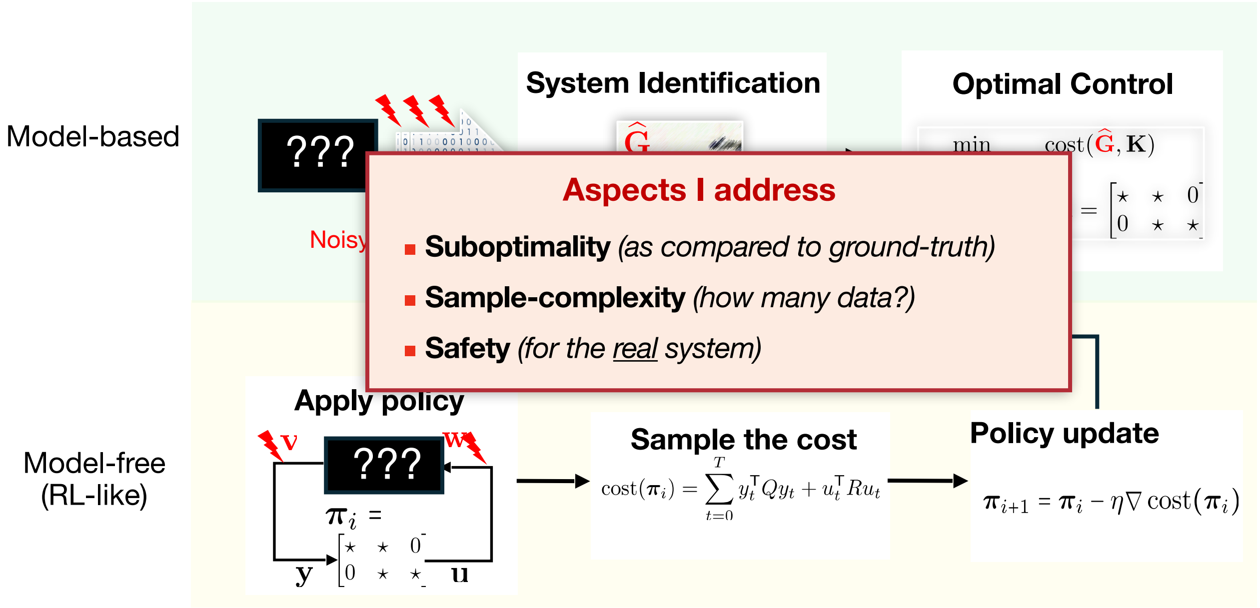

**Optimal Control**

$$\min_{\mathbf{K}\ \text{stabilizing}} \quad \text{cost}(\widehat{\mathbf{G}}, \mathbf{K})$$

$$\text{subject to} \quad \mathbf{K} = \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$$

Model-free
(RL-like)

# Learning-based control philosophies




## Aspects I address

- **Suboptimality** *(as compared to ground-truth)*
- **Sample-complexity** *(how many data?)*
- **Safety** *(for the real system)*

# Model-free learning of the globally optimal distributed LQ regulator[1]

[1] Furieri, Zheng, Kamgarpour, "*Learning the globally optimal distributed LQ regulator*", L4DC, 2020

## Result: Favorable optimization landscape[1]

The $\mathrm{cost}$ is nonconvex in distributed policies $\pi \in \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$. However;

# Model-free learning of the globally optimal distributed LQ regulator[1]

[1] Furieri, Zheng, Kamgarpour, "*Learning the globally optimal distributed LQ regulator*", L4DC, 2020

## Result: Favorable optimization landscape[1]

The $\mathrm{cost}$ is nonconvex in distributed policies $\pi \in \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$. However;

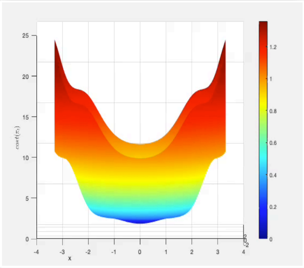
# Model-free learning of the globally optimal distributed LQ regulator[1]

[1] Furieri, Zheng, Kamgarpour, "*Learning the globally optimal distributed LQ regulator*", L4DC, 2020

## Result: Favorable optimization landscape[1]

The $\text{cost}$ is nonconvex in distributed policies $\pi \in \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$. However;

- Whenever QI holds, $\text{cost}(\pi)$ is Polyak-Lojasiewicz (PL)
- Several non-QI cases are also PL!
- … *PL holds for arbitrary sparsities?* No.



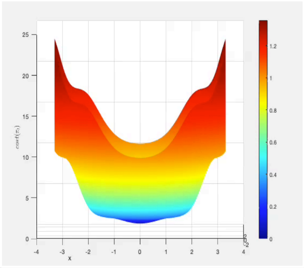
# Model-free learning of the globally optimal distributed LQ regulator[1]

[1] Furieri, Zheng, Kamgarpour, "*Learning the globally optimal distributed LQ regulator*", L4DC, 2020

## Result: Favorable optimization landscape[1]

The $\mathrm{cost}$ is nonconvex in distributed policies $\boldsymbol{\pi} \in \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix}$. However;

- Whenever QI holds, $\mathrm{cost}(\boldsymbol{\pi})$ is Polyak-Lojasiewicz (PL)
- Several non-QI cases are also PL!
- … *PL holds for arbitrary sparsities?* No.



## Result: Sample-complexity for global optimality[1]

Assume that PL holds. Run model-free policy gradient for $T$ steps with

$$T \propto \left( \mathrm{card}\left( \begin{bmatrix} \star & \star & 0 \\ 0 & \star & \star \end{bmatrix} \right)^2 \epsilon^{-2}\, \delta^{-4} \right)$$

Then: $\mathrm{cost}(\boldsymbol{\pi}_T) - \mathrm{cost}(\boldsymbol{\pi}^\star) \leq \epsilon$

with high probability $1 - \delta$.

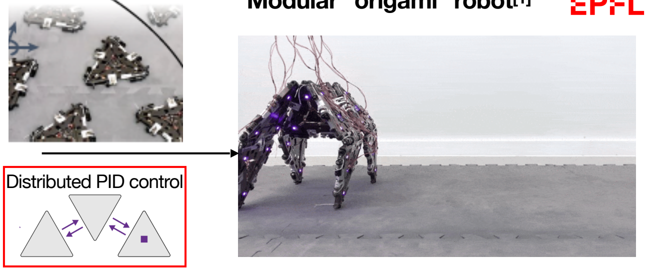
# Selected Contributions: Part 2

## Learning to control with stability guarantees for nonlinear systems

[1] Furieri, Galimberti, F. Trecate, “*Neural system level synthesis: learning over all stabilizing policies for nonlinear systems*”, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, “*Learning to boost the performance of stable nonlinear systems*”, [ArXiV, 2024]

- Complex real-world systems are nonlinear
- Frequent availability of stabilizing controllers around equilibrium or a reference

**Modular "origami" robot[1]**




**All-terrain legged-robots[2]**

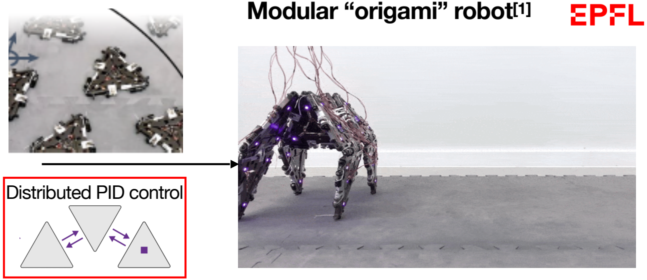


[1] Wisth, Camurri, Fallon, "*VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots*." IEEE Transactions on Robotics, 2022

[2] Belke, Holdcroft, Sigrist, Paik, "*Morphological flexibility in robotic systems through physical polygon meshing.*" Nature Machine Intelligence, 2023

- Complex real-world systems are nonlinear
- Frequent availability of stabilizing controllers around equilibrium or a reference

**Modular “origami” robot[1]**



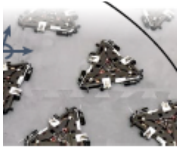

**All-terrain legged-robots[2]**

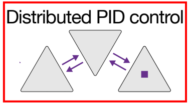

[1] Wisth, Camurri, Fallon, “*VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots*." IEEE Transactions on Robotics, 2022
[2] Belke, Holdcroft, Sigrist, Paik, "*Morphological flexibility in robotic systems through physical polygon meshing.*" Nature Machine Intelligence, 2023

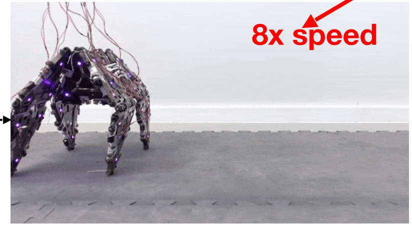
- Complex real-world systems are nonlinear
- Frequent availability of stabilizing controllers around equilibrium or a reference

**… however, stability is not enough**

**Modular "origami" robot[1]** EPFL





**8x speed**

**All-terrain legged-robots[2]** UNIVERSITY OF OXFORD

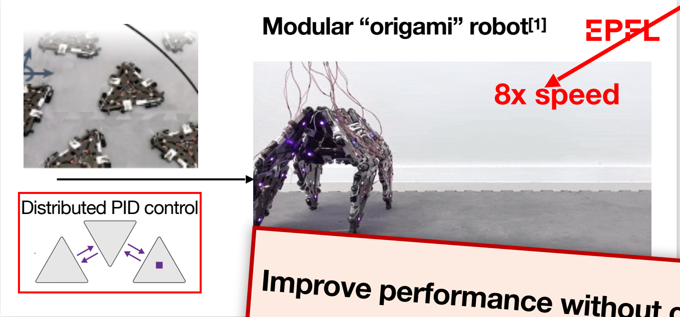

**>4x speed**

[1] Wisth, Camurri, Fallon, "*VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots*." IEEE Transactions on Robotics, 2022

[2] Belke, Holdcroft, Sigrist, Paik, "*Morphological flexibility in robotic systems through physical polygon meshing.*" Nature Machine Intelligence, 2023

- Complex real-world systems are nonlinear
- Frequent availability of stabilizing controllers around equilibrium or a reference

**… however, stability is not enough**





**Improve performance without compromising stability?**

[1] Wisth, Camurri, Fallon, "*VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots*." IEEE Transactions on Robotics, 2022

[2] Belke, Holdcroft, Sigrist, Paik, "*Morphological flexibility in robotic systems through physical polygon meshing.*" Nature Machine Intelligence, 2023

# Parametrization of all nonlinear stabilizing controllers[1]

[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]

$$\begin{cases} x_t = f_t(x_{t-1}, u_{t-1}) + w_t \\ u_t = K_t(x_{t:0}) \end{cases}$$

$$\mathbf{x} = (x_0, x_1, \ldots)$$
$$\mathbf{K}(\mathbf{x}) = (K_0(x_0), K_1(x_{1:0}), \ldots)$$

**Operator model**

$$\begin{cases} \mathbf{x} = \mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{w} \\ \mathbf{u} = \mathbf{K}(\mathbf{x}) \end{cases}$$

**Main result:**[1,2] Parametrization of robustly stabilizing controllers + completeness

# Parametrization of all nonlinear stabilizing controllers[1]

[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]

$$\begin{cases} x_t = f_t(x_{t-1}, u_{t-1}) + w_t \\ u_t = K_t(x_{t:0}) \end{cases}$$

$$\mathbf{x} = (x_0, x_1, \dots)$$
$$\mathbf{K}(\mathbf{x}) = (K_0(x_0), K_1(x_{1:0}), \dots)$$

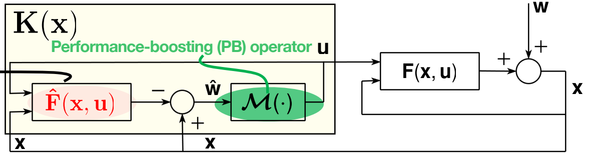
**Operator model**

$$\begin{cases} \mathbf{x} = \mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{w} \\ \mathbf{u} = \mathbf{K}(\mathbf{x}) \end{cases}$$

## Main result:[1,2] Parametrization of robustly stabilizing controllers + completeness

$\widehat{\mathbf{F}}(\mathbf{x}, \mathbf{u}) = (\mathbf{F} + \mathbf{\Delta})(\mathbf{x}, \mathbf{u})$

**Uncertain Internal model of the plant**




Assume the open-loop plant $\mathbf{x} = \mathcal{F}(\mathbf{u}, \mathbf{w})$ is stable, or pre-stabilized. Then:

# Parametrization of all nonlinear stabilizing controllers[1]

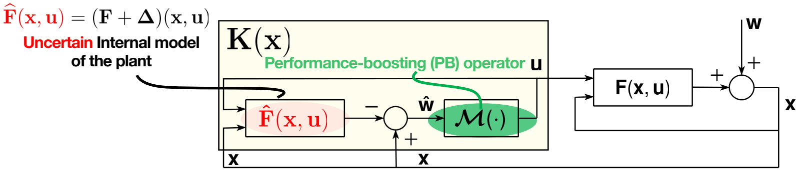
[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]

$$\begin{cases} x_t = f_t(x_{t-1}, u_{t-1}) + w_t \\ u_t = K_t(x_{t:0}) \end{cases} \xrightarrow[\mathbf{K}(\mathbf{x}) = (K_0(x_0), K_1(x_{1:0}), \ldots)]{\mathbf{x} = (x_0, x_1, \ldots)}$$

**Operator model**

$$\begin{cases} \mathbf{x} = \mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{w} \\ \mathbf{u} = \mathbf{K}(\mathbf{x}) \end{cases}$$

## Main result:[1,2] Parametrization of robustly stabilizing controllers + completeness




Assume the open-loop plant $\mathbf{x} = \mathcal{F}(\mathbf{u}, \mathbf{w})$ is stable, or pre-stabilized. Then:

- **Sufficiency:** the controller $\mathbf{K}(\mathbf{x})$ stabilizes the real system $\mathbf{F}(\mathbf{x},\mathbf{u})$ $\quad \text{gain}(\mathcal{M}) < \dfrac{1}{\text{gain}(\boldsymbol{\Delta})(\text{gain}(\mathcal{F})+1)}$

# Parametrization of all nonlinear stabilizing controllers[1]

[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]

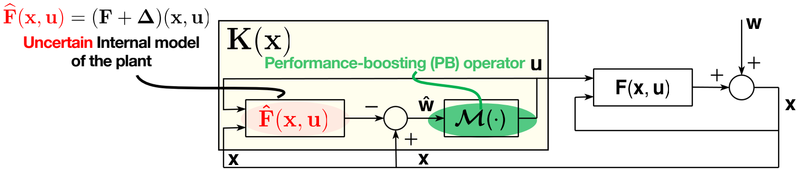
$$\begin{cases} x_t = f_t(x_{t-1}, u_{t-1}) + w_t \\ u_t = K_t(x_{t:0}) \end{cases}$$

$$\mathbf{x} = (x_0, x_1, \ldots)$$
$$\mathbf{K}(\mathbf{x}) = (K_0(x_0), K_1(x_{1:0}), \ldots)$$

**Operator model**

$$\begin{cases} \mathbf{x} = \mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{w} \\ \mathbf{u} = \mathbf{K}(\mathbf{x}) \end{cases}$$

## Main result:[1,2] Parametrization of robustly stabilizing controllers + completeness




Assume the open-loop plant $\mathbf{x} = \mathcal{F}(\mathbf{u}, \mathbf{w})$ is stable, or pre-stabilized. Then:

- **Sufficiency:** the controller $\mathbf{K}(\mathbf{x})$ stabilizes the <u>real</u> system $\mathbf{F}(\mathbf{x},\mathbf{u})$ $\quad \text{gain}(\mathcal{M}) < \dfrac{1}{\text{gain}(\mathbf{\Delta})(\text{gain}(\mathcal{F})+1)}$
- **Necessity:** if the system model is known $(\mathbf{\Delta} = 0)$, <u>*any stable closed-loop behavior*</u> is achieved by appropriately selecting a stable $\mathcal{M}$

# Parametrization of all nonlinear stabilizing controllers[1]

[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]

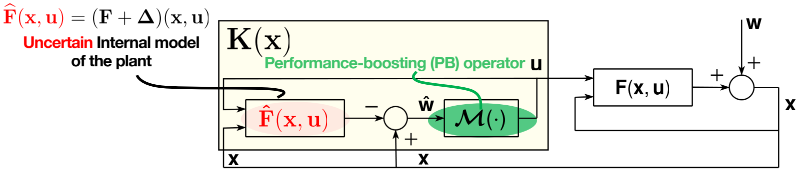
$$\begin{cases} x_t = f_t(x_{t-1}, u_{t-1}) + w_t \\ u_t = K_t(x_{t:0}) \end{cases}$$

$$\mathbf{x} = (x_0, x_1, \dots)$$
$$\mathbf{K}(\mathbf{x}) = (K_0(x_0), K_1(x_{1:0}), \dots)$$

**Operator model**

$$\begin{cases} \mathbf{x} = \mathbf{F}(\mathbf{x}, \mathbf{u}) + \mathbf{w} \\ \mathbf{u} = \mathbf{K}(\mathbf{x}) \end{cases}$$

## Main result:[1,2] Parametrization of robustly stabilizing controllers + completeness




Assume the open-loop plant $\mathbf{x} = \mathcal{F}(\mathbf{u}, \mathbf{w})$ is stable, or pre-stabilized. Then:

- **Sufficiency:** the controller $\mathbf{K}(\mathbf{x})$ stabilizes the real system $\mathbf{F}(\mathbf{x},\mathbf{u})$ if $\text{gain}(\mathcal{M}) < \dfrac{1}{\text{gain}(\boldsymbol{\Delta})(\text{gain}(\mathcal{F})+1)}$
- **Necessity:** if the system model is known $(\boldsymbol{\Delta} = 0)$, *any stable closed-loop behavior* is achieved by appropriately selecting a stable $\mathcal{M}$

# Importance of main result for nonlinear optimal control

[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]

$$\boxed{\begin{aligned} \max_{\mathbf{K} \text{ is stabilizing}} \quad & \mathbb{E}_{\mathbf{w}}\left[\text{performance}(\mathbf{x}, \mathbf{u})\right] \\ \text{subject to} \quad & x_t = f_t(x_{t-1}, u_{t-1}) + w_t, \\ & u_t = K_t(x_{t:0}), \quad t = 1, 2, \ldots \end{aligned}}$$

# Importance of main result for nonlinear optimal control


[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]

[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]


Nonlinear generalization of "Youla", "System Level Synthesis",…

$$
\begin{aligned}
\max_{\mathbf{K} \text{ is stabilizing}} \quad & \mathbb{E}_{\mathbf{w}}\left[\text{performance}(\mathbf{x},\mathbf{u})\right] \\
\text{subject to} \quad & x_t = f_t(x_{t-1}, u_{t-1}) + w_t, \\
& u_t = K_t(x_{t:0}), \quad t = 1, 2, \ldots
\end{aligned}
\quad \Longleftrightarrow \quad
\begin{aligned}
\max_{\mathcal{M} \text{ is stable}} \quad & \mathbb{E}_{\mathbf{w}}\left[\text{performance}(\mathbf{x},\mathbf{u})\right] \\
\text{subject to} \quad & x_t = f_t(x_{t-1}, u_{t-1}) + w_t, \\
& \hat{w}_t = x_t - f_t(x_{t-1}, u_{t-1}) \\
& u_t = \mathcal{M}_t(\hat{w}_{t:0}), \quad t = 1, 2, \ldots
\end{aligned}
$$

- **Theory:** directly optimize over stable closed-loop operator $\mathcal{M}$

# Importance of main result for nonlinear optimal control

[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]

Nonlinear generalization of "Youla", "System Level Synthesis",…

$$\boxed{\begin{aligned} \max_{\mathbf{K} \text{ is stabilizing}} \quad & \mathbb{E}_{\mathbf{w}}\left[\text{performance}(\mathbf{x}, \mathbf{u})\right] \\ \text{subject to} \quad & x_t = f_t(x_{t-1}, u_{t-1}) + w_t, \\ & u_t = K_t(x_{t:0}), \quad t = 1, 2, \ldots \end{aligned}} \iff \boxed{\begin{aligned} \max_{\mathcal{M} \text{ is stable}} \quad & \mathbb{E}_{\mathbf{w}}\left[\text{performance}(\mathbf{x}, \mathbf{u})\right] \\ \text{subject to} \quad & x_t = f_t(x_{t-1}, u_{t-1}) + w_t, \\ & \hat{w}_t = x_t - f_t(x_{t-1}, u_{t-1}) \\ & u_t = \mathcal{M}_t(\hat{w}_{t:0}), \quad t = 1, 2, \ldots \end{aligned}}$$

- **Theory:** directly optimize over stable closed-loop operator $\mathcal{M}$
- **Implementation:** compatible with **deep neural network** stable $\mathcal{M}$ operators![3]

TensorFlow
PyTorch

[3] *"Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness"*, Revay, Wang, Manchester, [TAC23]

# Importance of main result for nonlinear optimal control

[1] Furieri, Galimberti, F. Trecate, *"Neural system level synthesis: learning over all and only the stabilizing controllers for nonlinear systems"*, [CDC 2022]
[2] Furieri, Galimberti, F. Trecate, *"Learning to boost the performance of stable nonlinear systems"*, [ArXiV, 2024]

Nonlinear generalization of "Youla", "System Level Synthesis",…

$$\begin{aligned} \max_{\mathbf{K} \text{ is stabilizing}} \quad & \mathbb{E}_{\mathbf{w}}\left[\text{performance}(\mathbf{x}, \mathbf{u})\right] \\ \text{subject to} \quad & x_t = f_t(x_{t-1}, u_{t-1}) + w_t, \\ & u_t = K_t(x_{t:0}), \quad t = 1, 2, \ldots \end{aligned} \iff \begin{aligned} \max_{\mathcal{M} \text{ is stable}} \quad & \mathbb{E}_{\mathbf{w}}\left[\text{performance}(\mathbf{x}, \mathbf{u})\right] \\ \text{subject to} \quad & x_t = f_t(x_{t-1}, u_{t-1}) + w_t, \\ & \hat{w}_t = x_t - f_t(x_{t-1}, u_{t-1}) \\ & u_t = \mathcal{M}_t(\hat{w}_{t:0}), \quad t = 1, 2, \ldots \end{aligned}$$

- **Theory:** directly optimize over stable closed-loop operator $\mathcal{M}$
- **Implementation:** compatible with **deep neural network** stable $\mathcal{M}$ operators![3]

TensorFlow
PyTorch

[3] *"Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness"*, Revay, Wang, Manchester, [TAC23]

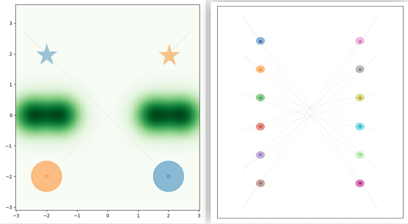
## Remark: results compatible with *distributed* nonlinear control

[4] Massai, Saccani, Furieri, F. Trecate, «*Unconstrained learning of* ***networked nonlinear systems*** *via free parametrization of stable interconnected operators*», [ECC24]

[5] Saccani, Massai, Furieri, F. Trecate, «***Optimal distributed control*** *with stability guarantees by training a network of neural closed-loop maps», [ArXiV 2024]*

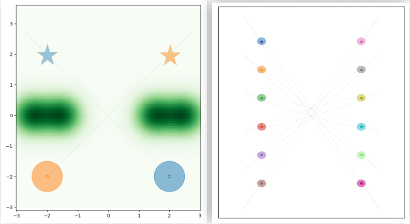
# Numerical experiments: mobile robots

**…before performance-boosting**



# Numerical experiments: mobile robots

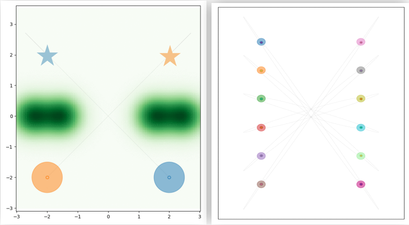
**…before performance-boosting**



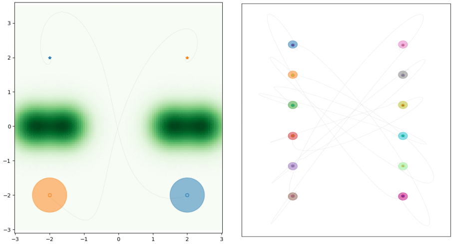
# Numerical experiments: mobile robots

$$\mathrm{cost}(\mathbf{x},\mathbf{u}) = \mathrm{cost}_{\mathrm{target}}(\mathbf{x},\mathbf{u}) + \mathrm{cost}_{\mathrm{collisions}}(\mathbf{x})$$

**…before performance-boosting**



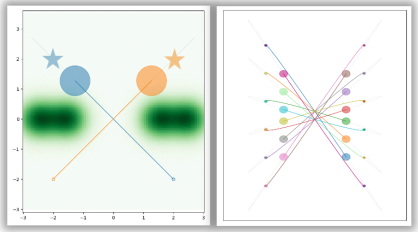
**After performance-boosting**



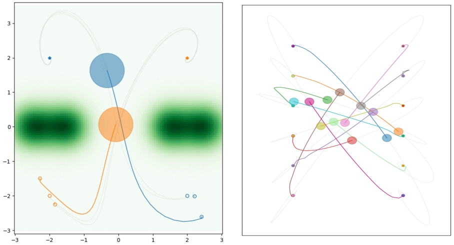
# Numerical experiments: mobile robots

$$\text{cost}(\mathbf{x},\mathbf{u}) = \text{cost}_{\text{target}}(\mathbf{x},\mathbf{u}) + \text{cost}_{\text{collisions}}(\mathbf{x})$$

**...before performance-boosting**

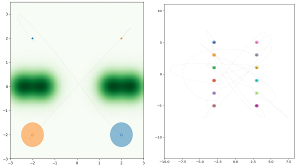


**After performance-boosting**



**CL stability guaranteed even with partial training**

25% training



75% training