Learning with Guarantees

Parametrising Neural Feedback Policies with Stability & Robustness Certificates

Nicholas H. Barbara

Doctor of Philosophy – Defence

20 October 2025

THE UNIVERSITY OF SYDNEY
Australian Centre for Robotics

Outline

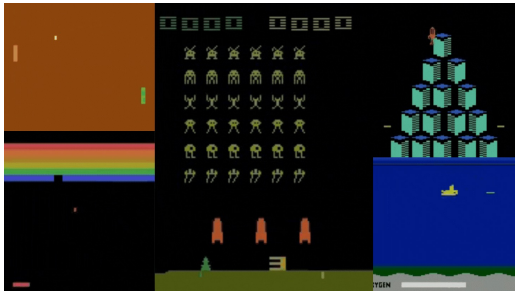
1. Motivation
2. Background: robust neural networks
3. Lipschitz-bounded policy networks
4. React to surprises with Youla-REN (Theory)
5. React to surprises with Youla-REN (Application)
6. Robust recurrent deep networks
7. Conclusions

Outline

1. Motivation
2. Background: robust neural networks
3. Lipschitz-bounded policy networks
4. React to surprises with Youla-REN (Theory)
5. React to surprises with Youla-REN (Application)
6. Robust recurrent deep networks
7. Conclusions

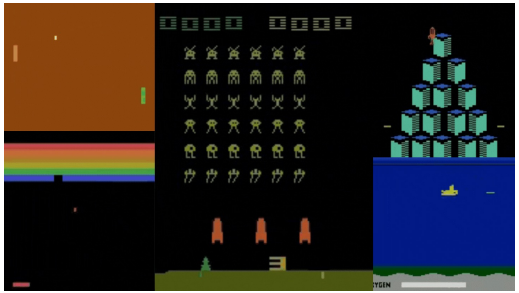
Deep Reinforcement Learning (RL)

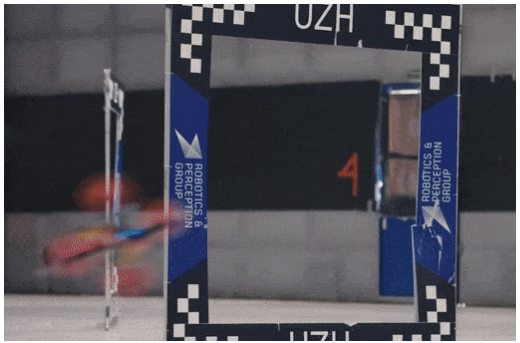
Deep Reinforcement Learning (RL)

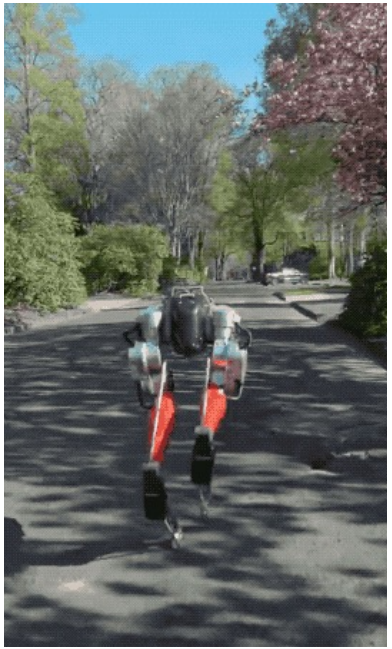


[1] Mnih et al., Human-level control through deep reinforcement learning. *Nature* (2015).

Deep Reinforcement Learning (RL)





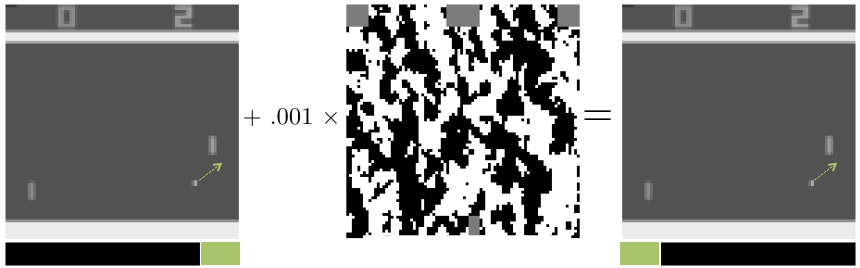


[1] Mnih et al., Human-level control through deep reinforcement learning. *Nature* (2015).
[2] Foehn et al., Agilicious: Open-Source and Open-Hardware Agile Quadrotor for Vision-Based Flight. *Science Robotics* (2022).
[3] Miki et al., Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics* (2022).
[4] Siekmann et al., Sim-to-real learning of all common bipedal gaits via periodic reward composition. *ICRA* (2021).

No Closed-Loop Guarantees

No Closed-Loop Guarantees

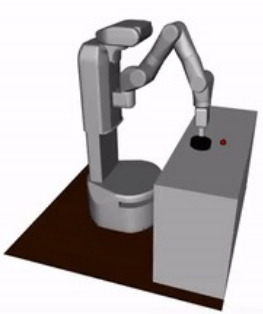
- Deep RL is not robust to small perturbations



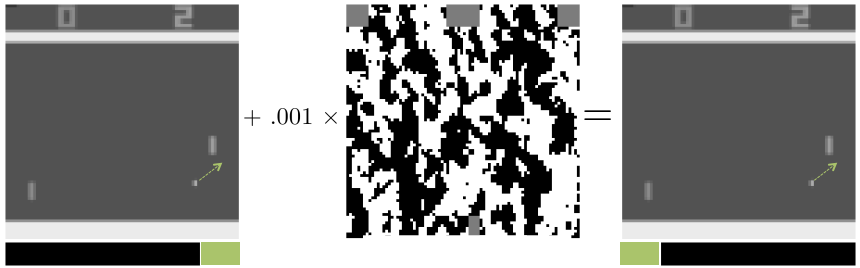
[5] Huang et al., Adversarial attacks on neural network policies. *ICLR* (2017).

No Closed-Loop Guarantees

- Deep RL is not robust to small perturbations
- Can improve robustness with empirical methods



https://youtu.be/2AGQolc8CiQ?t=50



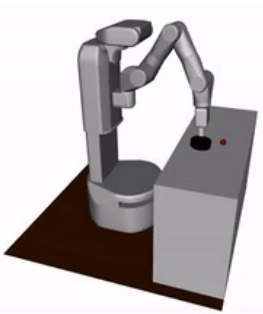
action taken: **down**
original input

action taken: **noop**
adversarial input

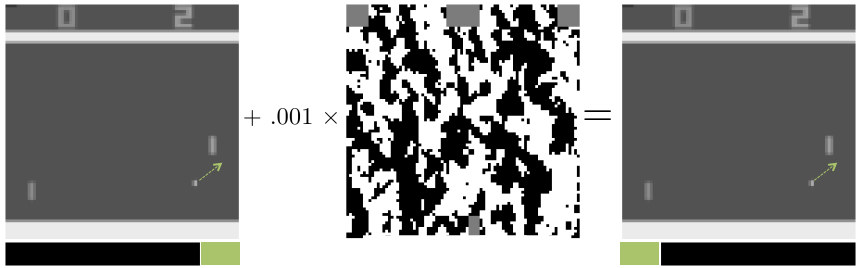
[5] Huang et al., Adversarial attacks on neural network policies. *ICLR* (2017).
[6] Peng et al., Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *ICRA* (2018).

No Closed-Loop Guarantees

- Deep RL is not robust to small perturbations
- Can improve robustness with empirical methods
- …but without *guarantees*, still closed-loop unstable!



https://youtu.be/2AGQolc8CiQ?t=50



action taken: **down**
original input

action taken: **noop**
adversarial input

[5] Huang et al., Adversarial attacks on neural network policies. *ICLR* (2017).
[6] Peng et al., Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *ICRA* (2018).
[7] Shi et al., Rethinking Robustness Assessment: Adversarial Attacks on Learning-based Quadrupedal Locomotion Controllers. *Robotics: Science and Systems* (2024).

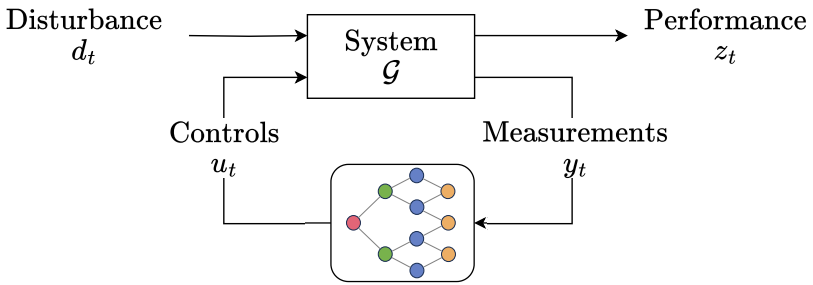
Black-Box Methods

Black-Box Methods

- Deep RL relies on black-box Neural Networks (NNs)

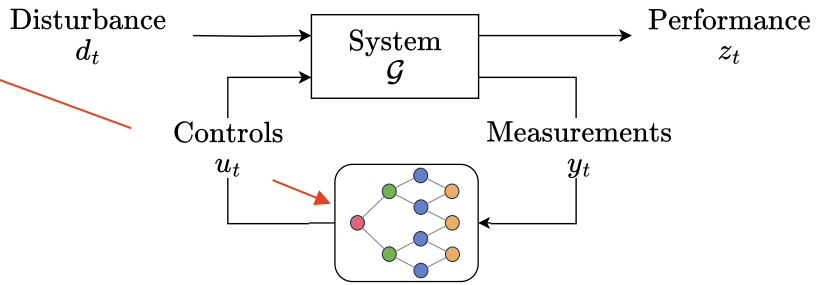
Black-Box Methods

- Deep RL relies on black-box Neural Networks (NNs)



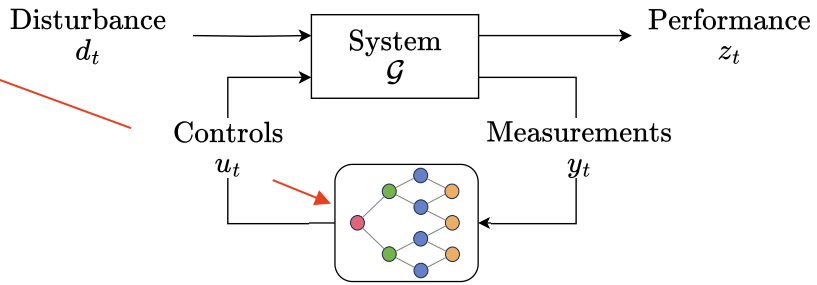
Black-Box Methods

- Deep RL relies on black-box Neural Networks (NNs)



Black-Box Methods

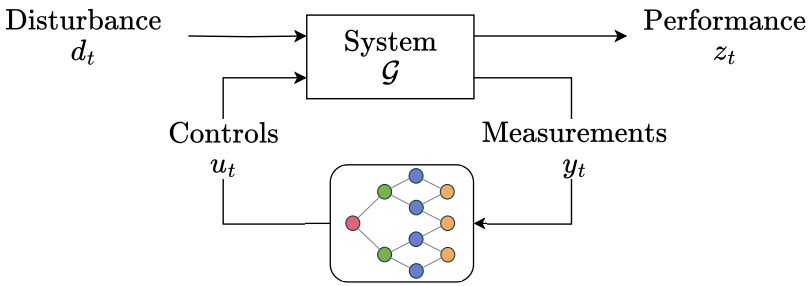
- Deep RL relies on black-box Neural Networks (NNs)
- Problem:
  - No prior system knowledge in the policy architecture
  - No prior system knowledge in the policy parametrisation
- …so closed-loop stability is *not explicitly considered!*



Question

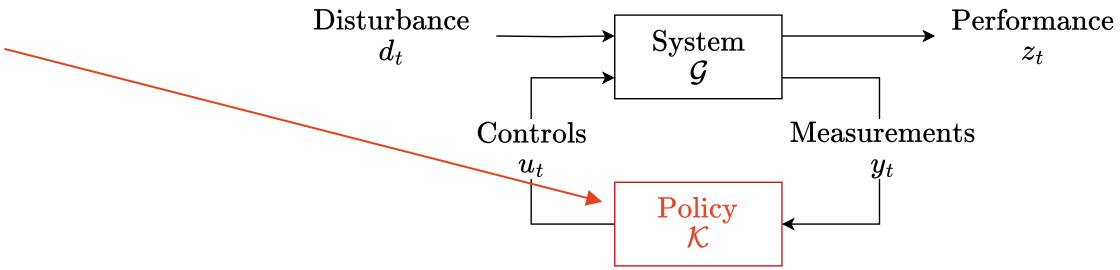
Can we learn robustly-stabilising neural feedback policies with closed-loop guarantees for nonlinear systems?

Stable-by-Design Neural Feedback Policies



Stable-by-Design Neural Feedback Policies

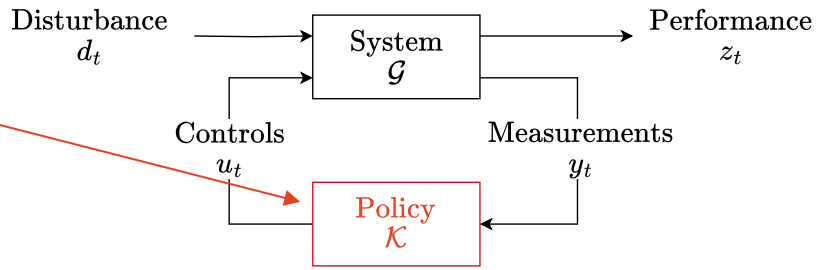
Construct a new *policy class*:



Stable-by-Design Neural Feedback Policies

Construct a new *policy class*:

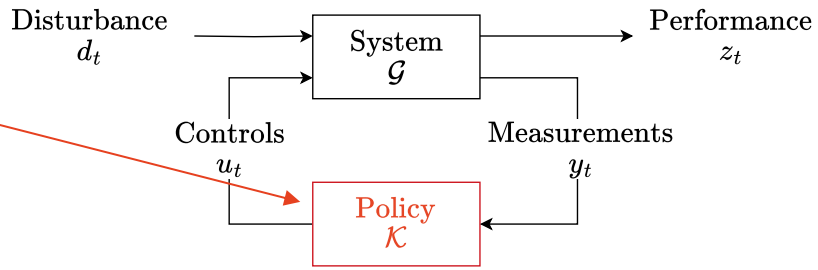
1. Closed-loop robust stability by construction
2. Differentiable, scalable parametrisation
3. Expressive parametrisation



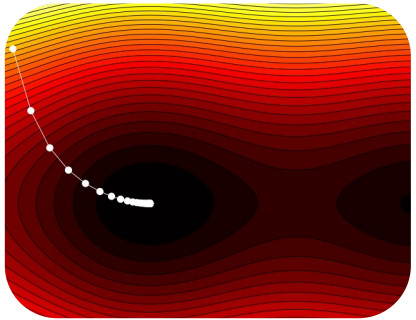
Stable-by-Design Neural Feedback Policies

Construct a new *policy class*:

1. Closed-loop robust stability by construction
2. Differentiable, scalable parametrisation
3. Expressive parametrisation



Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



Outline

1. Motivation
2. Background: robust neural networks
3. Lipschitz-bounded policy networks
4. React to surprises with Youla-REN (Theory)
5. React to surprises with Youla-REN (Application)
6. Robust recurrent deep networks
7. Conclusions

Robust Neural Networks

Robust Neural Networks

- Problems are caused by black-box NNs
- Sensitive to changes in their inputs, issues with internal instability

Robust Neural Networks

- Problems are caused by black-box NNs
- Sensitive to changes in their inputs, issues with internal instability

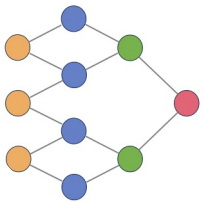
- Solution: use *robust neural networks*[8]

[8] Manchester, Wang, & Barbara, Neural Networks in the Loop: Learning with Stability and Robustness Guarantees, *accepted to Ann. Rev. Control, Robotics, Autonomous Systems* (2025).

Robust Neural Networks

- Problems are caused by black-box NNs
- Sensitive to changes in their inputs, issues with internal instability

- Solution: use *robust neural networks*[8]



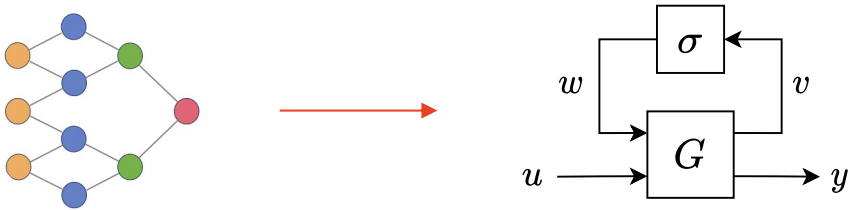
Neural networks

[8] Manchester, Wang, & Barbara, Neural Networks in the Loop: Learning with Stability and Robustness Guarantees, *accepted to Ann. Rev. Control, Robotics, Autonomous Systems* (2025).

Robust Neural Networks

- Problems are caused by black-box NNs
- Sensitive to changes in their inputs, issues with internal instability

- Solution: use *robust neural networks*[8]



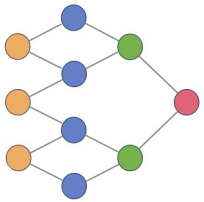
Neural networks + Robust control theory

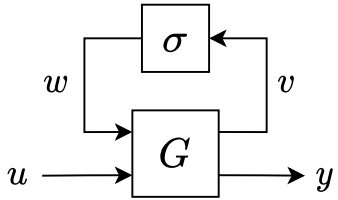
[8] Manchester, Wang, & Barbara, Neural Networks in the Loop: Learning with Stability and Robustness Guarantees, *accepted to Ann. Rev. Control, Robotics, Autonomous Systems* (2025).

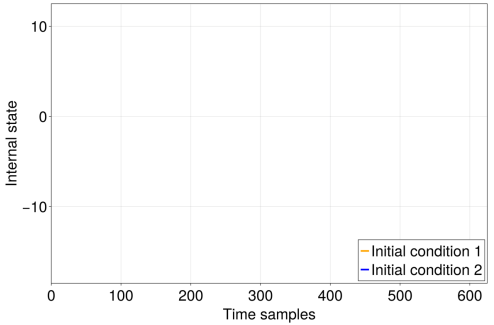
Robust Neural Networks

- Problems are caused by black-box NNs
- Sensitive to changes in their inputs, issues with internal instability

- Solution: use *robust neural networks*[8]



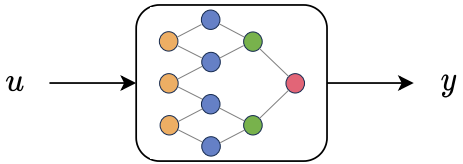




Neural networks + Robust control theory = Provable guarantees of smooth, stable response

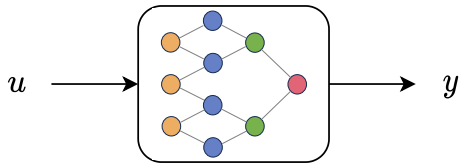
[8] Manchester, Wang, & Barbara, Neural Networks in the Loop: Learning with Stability and Robustness Guarantees, *accepted to Ann. Rev. Control, Robotics, Autonomous Systems* (2025).

Input-Output Robustness: Lipschitz Continuity



Input-Output Robustness: Lipschitz Continuity

- Lipschitz systems respond smoothly to changes in their inputs:

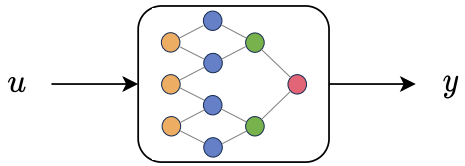


$$\|\Delta y\|_2 \leq \gamma \|\Delta u\|_2, \quad \gamma \geq 0$$

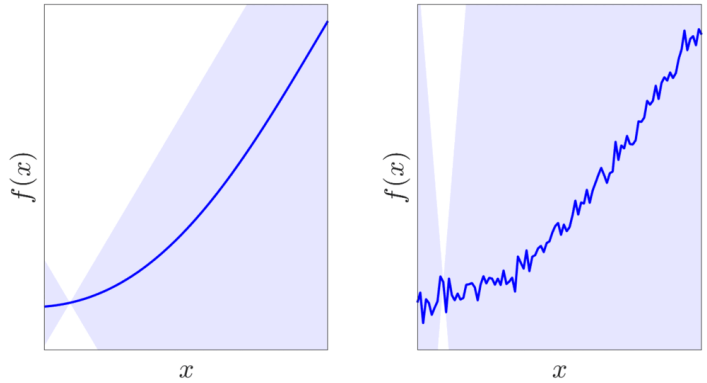
Input-Output Robustness: Lipschitz Continuity

- Lipschitz systems respond smoothly to changes in their inputs:

$$\|\Delta y\|_2 \leq \gamma \|\Delta u\|_2, \quad \gamma \geq 0$$



- Small Lipschitz constant ➔ smooth response
- Large Lipschitz constant ➔ sensitive to perturbations

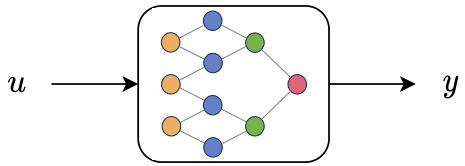


[9] Pauli et al., Neural Network Training Under Semidefinite Constraints, *CDC* (2022).

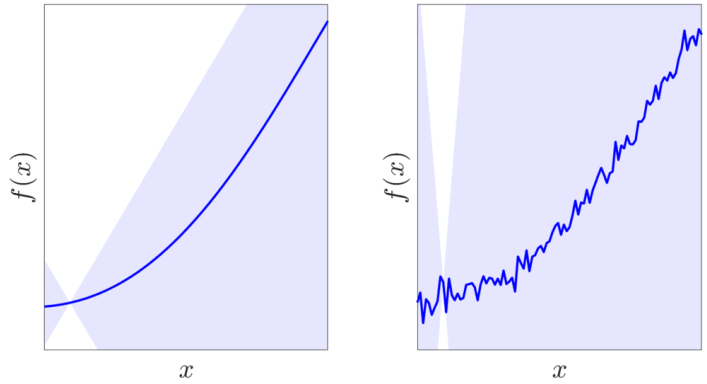
Input-Output Robustness: Lipschitz Continuity

- Lipschitz systems respond smoothly to changes in their inputs:

$$\|\Delta y\|_2 \leq \gamma \|\Delta u\|_2, \quad \gamma \geq 0$$

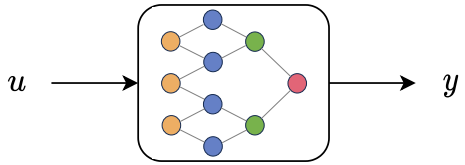


- Small Lipschitz constant ➔ smooth response
- Large Lipschitz constant ➔ sensitive to perturbations
- Generalise to other I/O properties with IQCs[10]

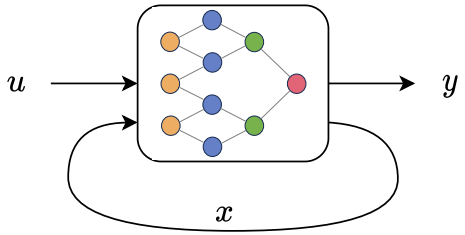


[9] Pauli et al., Neural Network Training Under Semidefinite Constraints, *CDC* (2022).
[10] Megretski & Rantzer, System Analysis via Integral Quadratic Constraints, *TAC* (1997).

Internal Stability: Contraction

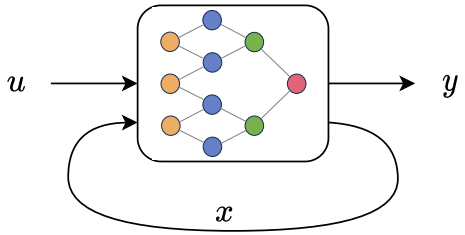


Internal Stability: Contraction



Internal Stability: Contraction

- Contracting systems forget their initial conditions exponentially



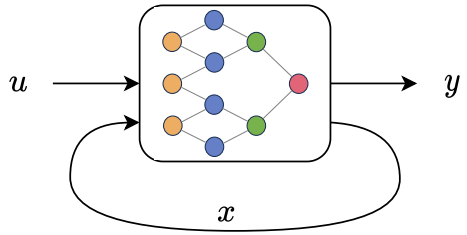
Internal Stability: Contraction

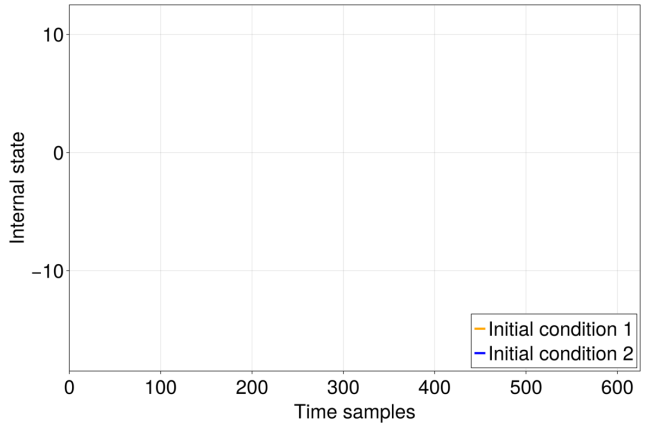
- Contracting systems forget their initial conditions exponentially

- For a discrete-time contracting system ($\Delta u = 0$)

$$|\Delta x_t| \leq \beta \alpha^t |\Delta x_0|, \quad \alpha \in [0, 1), \ \beta \geq 0$$

- I.e., incremental exponential stability





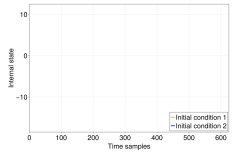
[11] Lohmiller & Slotine, On Contraction Analysis for Nonlinear Systems. *Automatica* (1998).

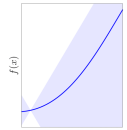
Stable & Robust Neural Networks

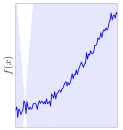
- Our suite of robust neural networks @ the ACFR[8]:

[8] Manchester, Wang, & Barbara, Neural Networks in the Loop: Learning with Stability and Robustness Guarantees, *accepted to Ann. Rev. Control Robotics & Autonomous Systems.* (2025).

Stable & Robust Neural Networks



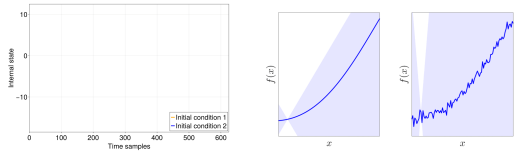




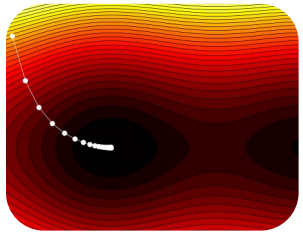
- Our suite of robust neural networks @ the ACFR[8]:
  - Static (feedforward) and dynamic (recurrent) models
  - Built-in guarantees of incremental stability (contraction) & robustness (Lipschitz, IQC)

[8] Manchester, Wang, & Barbara, Neural Networks in the Loop: Learning with Stability and Robustness Guarantees, *accepted to Ann. Rev. Control Robotics & Autonomous Systems.* (2025).

Stable & Robust Neural Networks

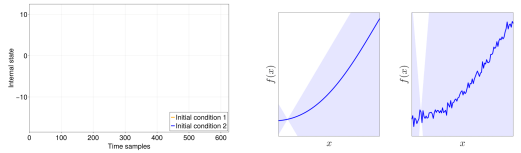


- Our suite of robust neural networks @ the ACFR[8]:
  - Static (feedforward) and dynamic (recurrent) models
  - Built-in guarantees of incremental stability (contraction) & robustness (Lipschitz, IQC)
  - **Directly parametrised** to allow unconstrained optimisation
  - Universal approximators of all contracting & Lipschitz systems

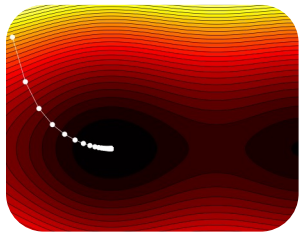


[8] Manchester, Wang, & Barbara, Neural Networks in the Loop: Learning with Stability and Robustness Guarantees, *accepted to Ann. Rev. Control Robotics & Autonomous Systems.* (2025).

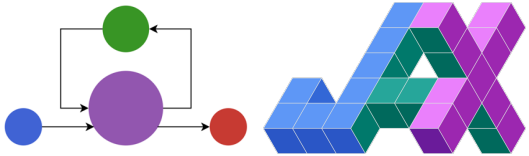
Stable & Robust Neural Networks



- Our suite of robust neural networks @ the ACFR[8]:
  - Static (feedforward) and dynamic (recurrent) models
  - Built-in guarantees of incremental stability (contraction) & robustness (Lipschitz, IQC)
  - **Directly parametrised** to allow unconstrained optimisation
  - Universal approximators of all contracting & Lipschitz systems



- Open-source and ready for use in Julia and Python (JAX)[8,12]



[8] Manchester, Wang, & Barbara, Neural Networks in the Loop: Learning with Stability and Robustness Guarantees, *accepted to Ann. Rev. Control Robotics & Autonomous Systems.* (2025).
[12] Barbara et al., RobustNeuralNetworks.jl: a Package for Machine Learning and Data-Driven Control with Certified Robustness, *JuliaCon Proceedings* (2025).

Examples of Robust Neural Networks

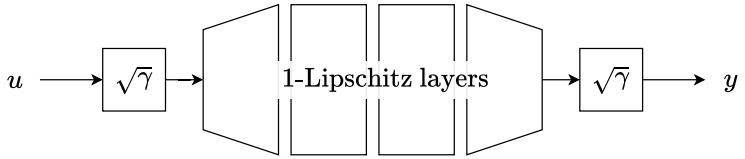
Two main robust NN architectures used in this talk:

Examples of Robust Neural Networks

Two main robust NN architectures used in this talk:

1. Sandwich network[13]

   - Static, feedforward DNNs (MLP, CNN, etc…)
   - Certified Lipschitz bound

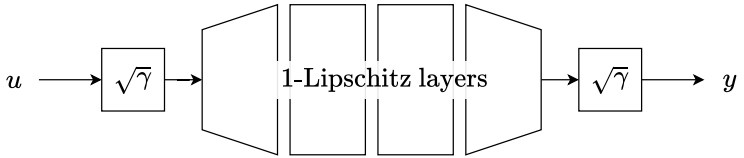


[13] Wang & Manchester, Direct Parameterization of Lipschitz-Bounded Deep Networks, *ICML* (2023).

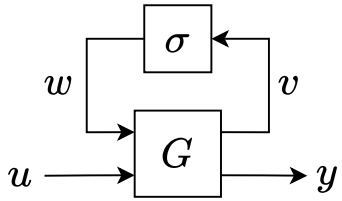
Examples of Robust Neural Networks

Two main robust NN architectures used in this talk:

1. Sandwich network[13]
   - Static, feedforward DNNs (MLP, CNN, etc…)
   - Certified Lipschitz bound



2. Recurrent Equilibrium Network (REN)[14]
   - Recurrent / dynamic models
   - Certified contraction and Lipschitz bounds
   - Universal approximator for contracting & Lipschitz systems



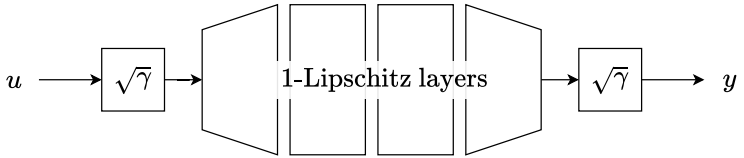
[13] Wang & Manchester, Direct Parameterization of Lipschitz-Bounded Deep Networks, *ICML* (2023).
[14] Revay, Wang, & Manchester, Recurrent Equilibrium Networks: Flexible Dynamic Models with Guaranteed Stability and Robustness, *TAC (2023).*

Examples of Robust Neural Networks

Two main robust NN architectures used in this talk:

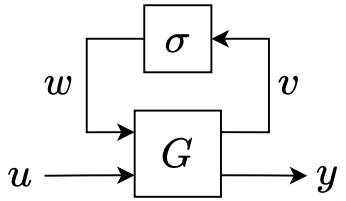
1. Sandwich network[13]

   - Static, feedforward DNNs (MLP, CNN, etc…)
   - Certified Lipschitz bound



2. Recurrent Equilibrium Network (REN)[14]

   - Recurrent / dynamic models
   - Certified contraction and Lipschitz bounds
   - Universal approximator for contracting & Lipschitz systems



For more, see Manchester et al., *Ann. Rev. Control, Robotics, Autonomous Systems* (2025).

[13] Wang & Manchester, Direct Parameterization of Lipschitz-Bounded Deep Networks, *ICML* (2023).
[14] Revay, Wang, & Manchester, Recurrent Equilibrium Networks: Flexible Dynamic Models with Guaranteed Stability and Robustness, *TAC (2023).*

Outline

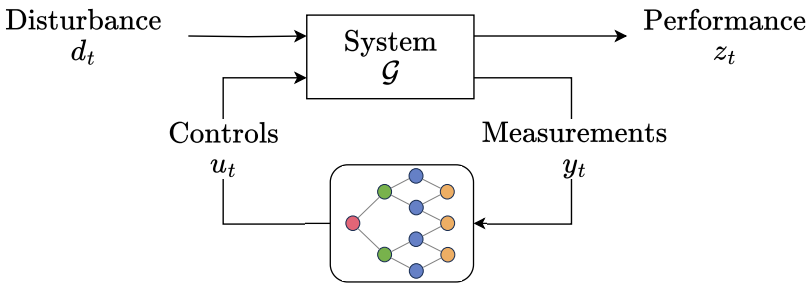
1. Motivation
2. Background: robust neural networks
3. Lipschitz-bounded policy networks
4. React to surprises with Youla-REN (Theory)
5. React to surprises with Youla-REN (Application)
6. Robust recurrent deep networks
7. Conclusions

Lipschitz-Bounded Policy Networks

Construct a new *policy class*:

1. Closed-loop robust stability by construction
2. Differentiable, scalable parametrisation
3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



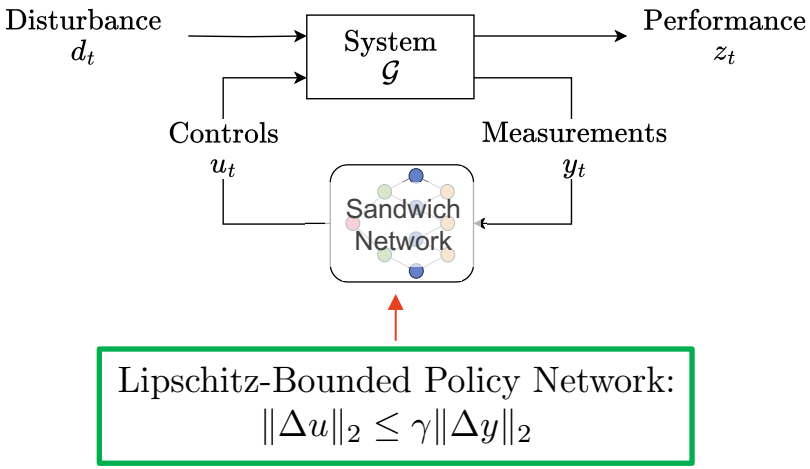
[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Lipschitz-Bounded Policy Networks

Construct a new *policy class*:

1. Closed-loop robust stability by construction
2. Differentiable, scalable parametrisation
3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Lipschitz-Bounded Policy Networks

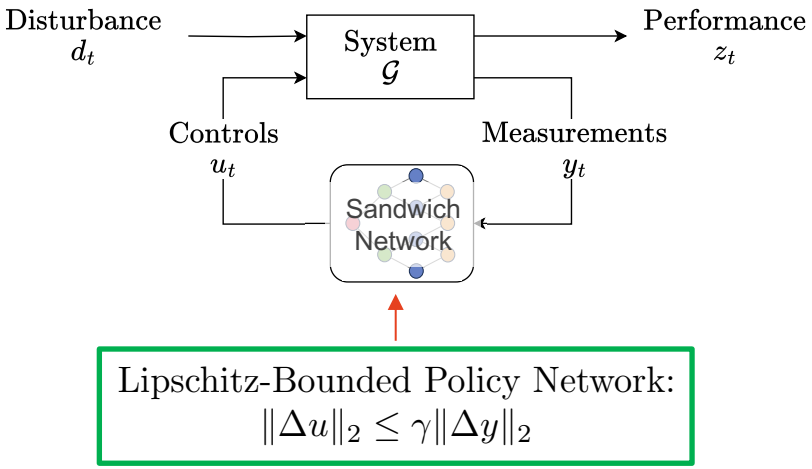
Construct a new *policy class*:

❌ 1. Closed-loop robust stability by construction

✅ 2. Differentiable, scalable parametrisation

✅ 3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Lipschitz-Bounded Policy Networks

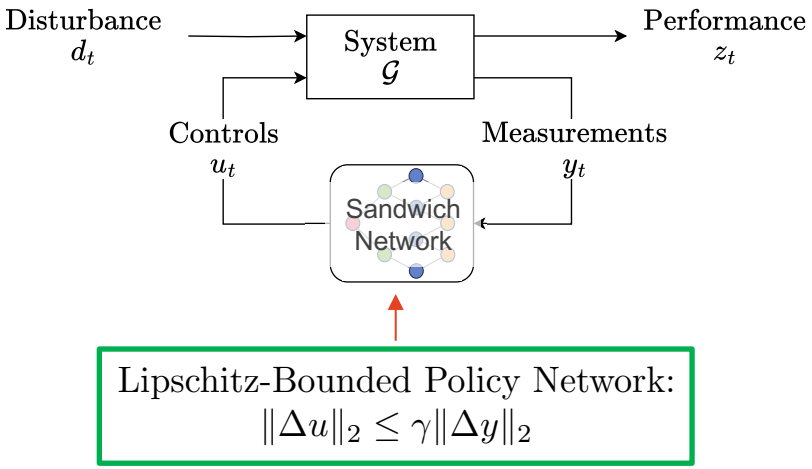
Construct a new *policy class*:

✅ 1. Robust to perturbations by construction

✅ 2. Differentiable, scalable parametrisation

✅ 3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Lipschitz-Bounded Policy Networks

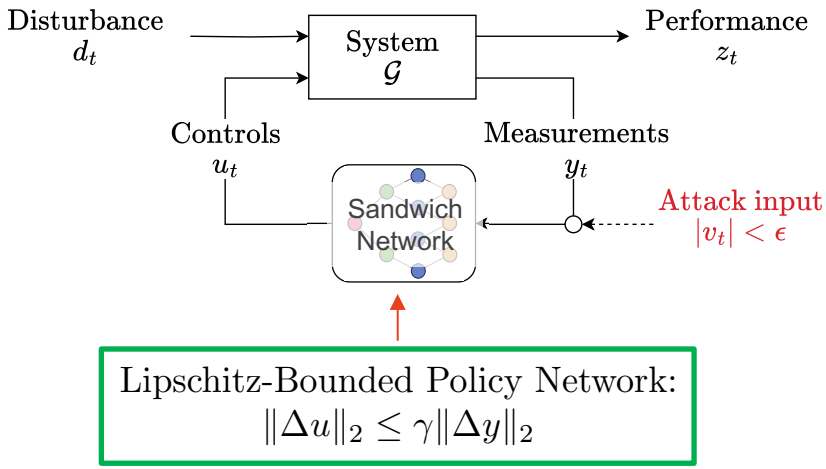
Construct a new *policy class*:

✅ 1. Robust to perturbations by construction

✅ 2. Differentiable, scalable parametrisation

✅ 3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Lipschitz-Bounded Policy Networks

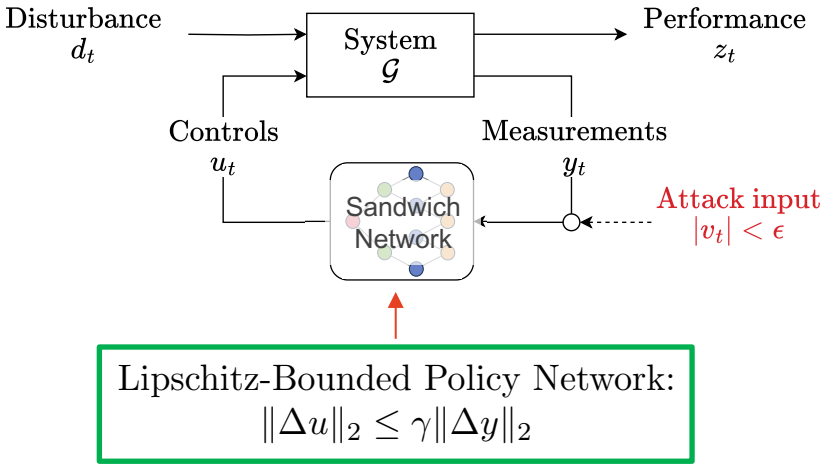
Construct a new *policy class*:

✅ 1. Robust to perturbations by construction

✅ 2. Differentiable, scalable parametrisation

✅ 3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



$$u = \mathcal{K}(y), \quad u' = \mathcal{K}(y + v)$$

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Lipschitz-Bounded Policy Networks

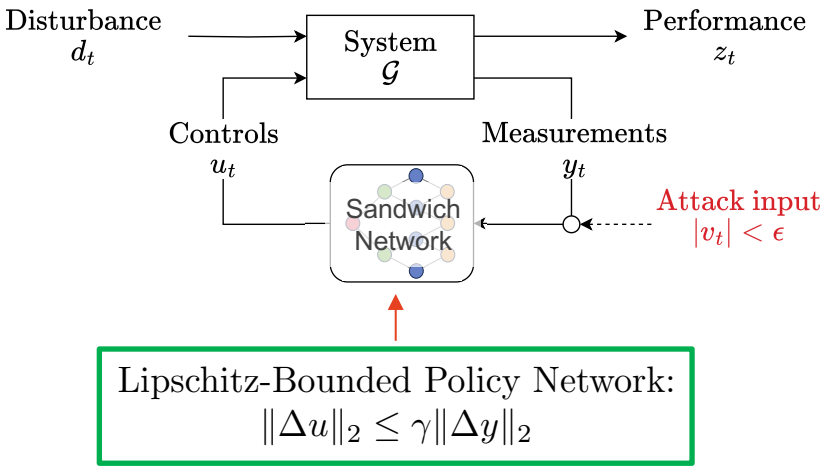
Construct a new *policy class*:

✅ 1. Robust to perturbations by construction

✅ 2. Differentiable, scalable parametrisation

✅ 3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



$$u = \mathcal{K}(y), \quad u' = \mathcal{K}(y + v)$$

$$\|u' - u\|_2 \leq \gamma \|v\|_2$$

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Playing Atari Pong (Noise-Free)

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Playing Atari Pong (Noise-Free)

Trained RL Agent

Unconstrained CNN

Lipschitz-Bounded CNN

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Playing Atari Pong (Noise-Affected)

Unconstrained CNN

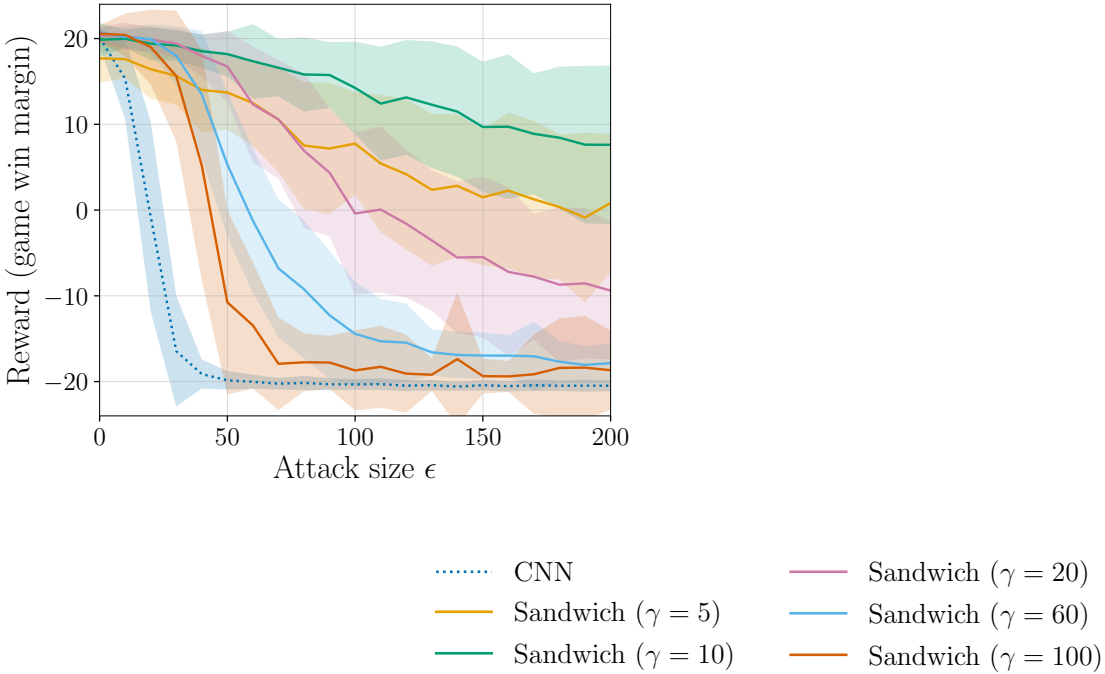
Lipschitz-Bounded CNN

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Tuneable Robustness to Adversarial Attacks

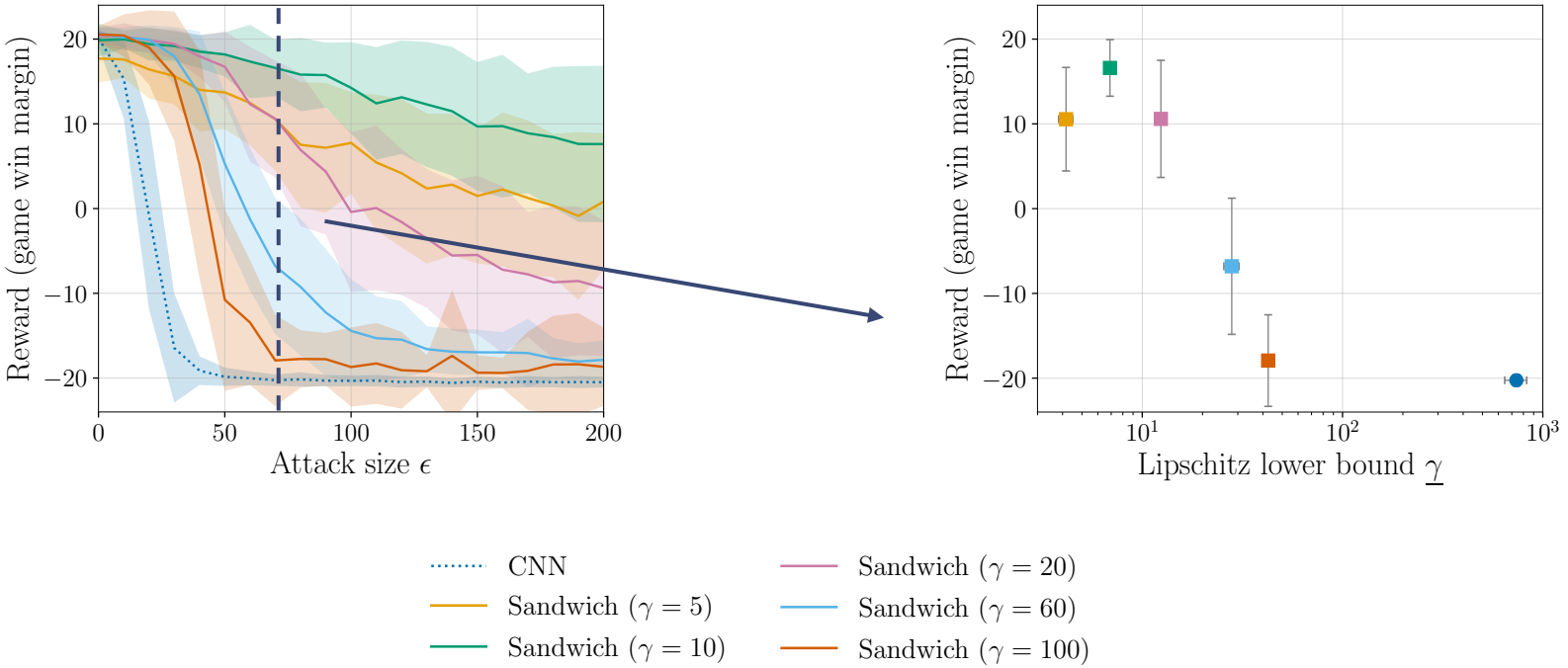
[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Tuneable Robustness to Adversarial Attacks



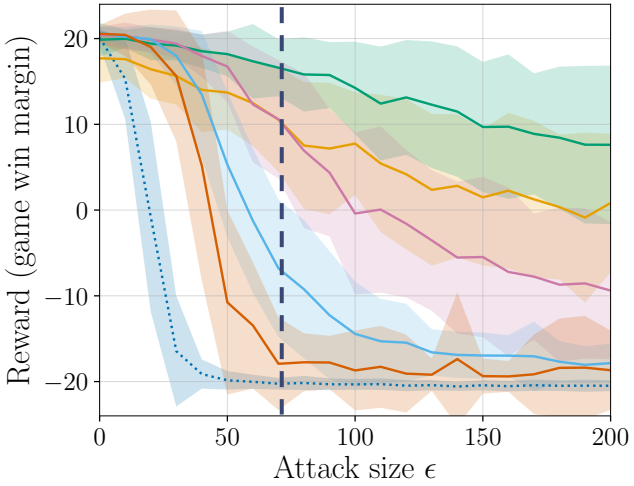
[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

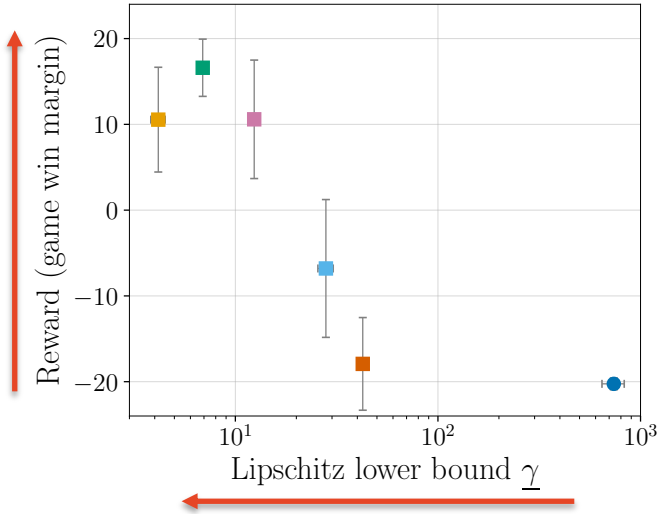
Tuneable Robustness to Adversarial Attacks



[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Tuneable Robustness to Adversarial Attacks

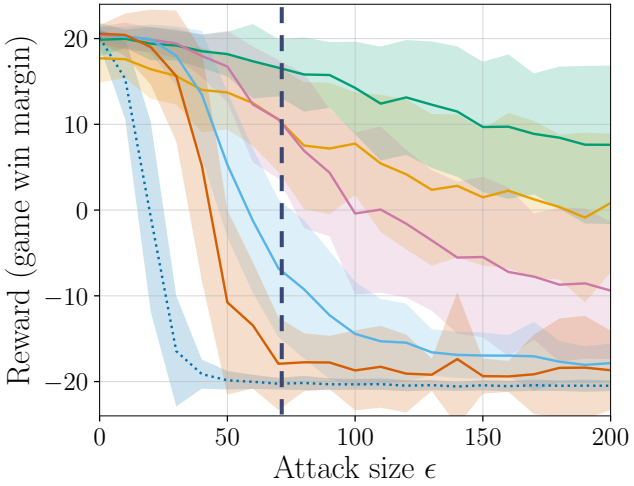


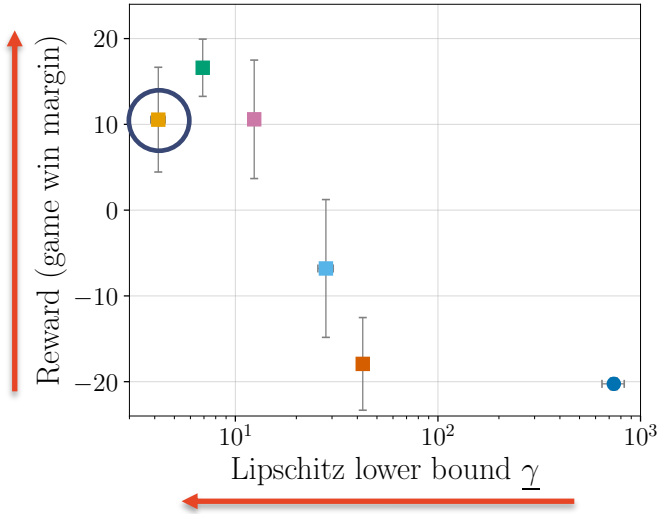


CNN
Sandwich ($\gamma = 5$)
Sandwich ($\gamma = 10$)
Sandwich ($\gamma = 20$)
Sandwich ($\gamma = 60$)
Sandwich ($\gamma = 100$)

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Tuneable Robustness to Adversarial Attacks

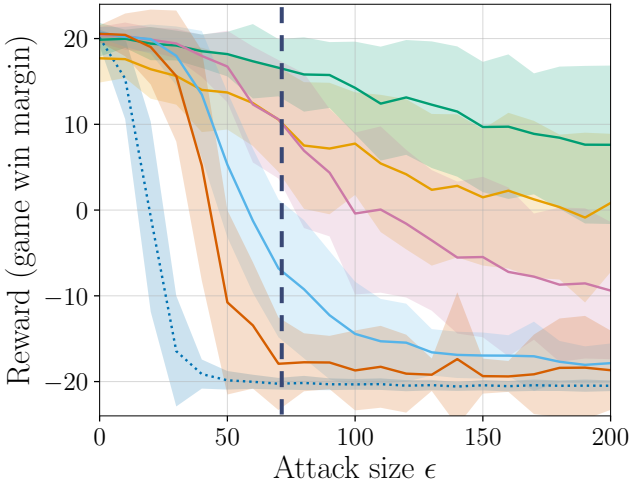


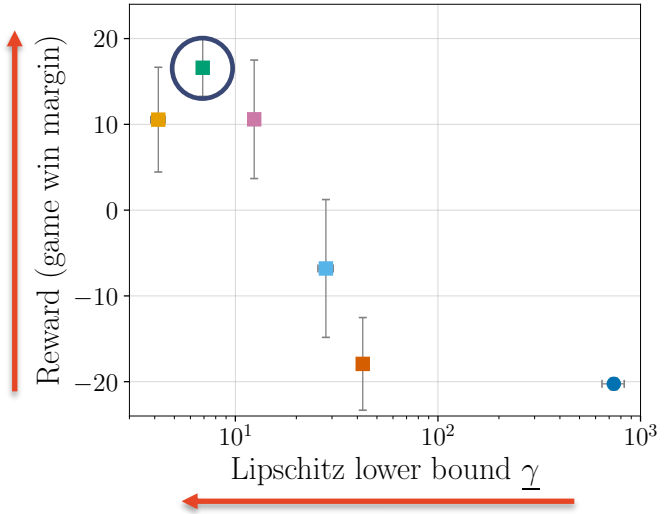


CNN
Sandwich ($\gamma = 5$)
Sandwich ($\gamma = 10$)
Sandwich ($\gamma = 20$)
Sandwich ($\gamma = 60$)
Sandwich ($\gamma = 100$)

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Tuneable Robustness to Adversarial Attacks

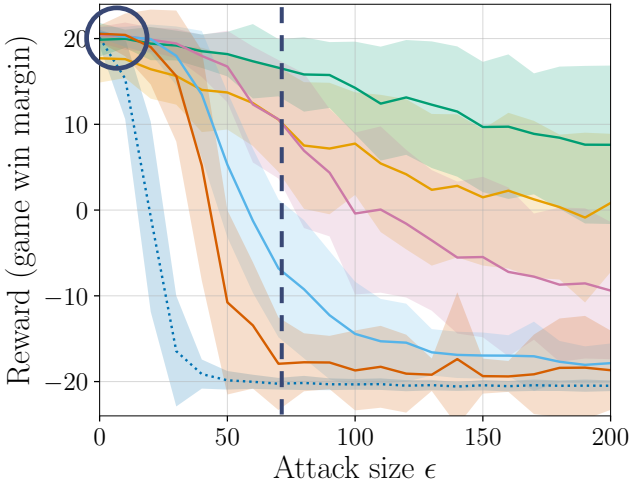


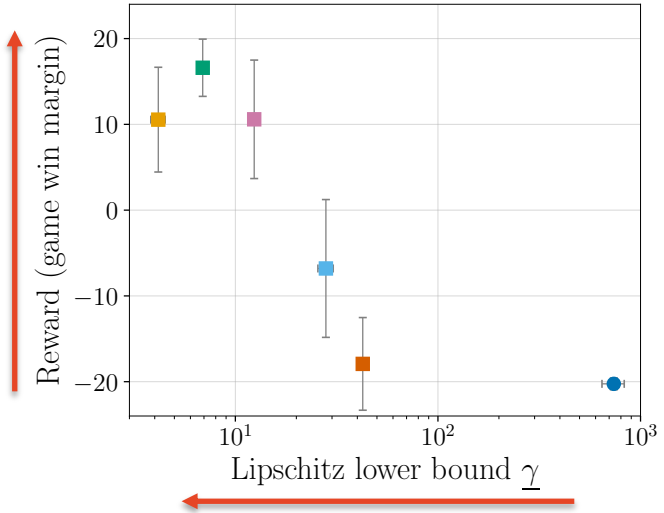


CNN
Sandwich ($\gamma = 5$)
Sandwich ($\gamma = 10$)
Sandwich ($\gamma = 20$)
Sandwich ($\gamma = 60$)
Sandwich ($\gamma = 100$)

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Tuneable Robustness to Adversarial Attacks





CNN
Sandwich ($\gamma = 5$)
Sandwich ($\gamma = 10$)
Sandwich ($\gamma = 20$)
Sandwich ($\gamma = 60$)
Sandwich ($\gamma = 100$)

[15] Barbara, Wang, & Manchester, On Robust Reinforcement Learning with Lipschitz-Bounded Policy Networks, *SysDO* (2024).

Outline

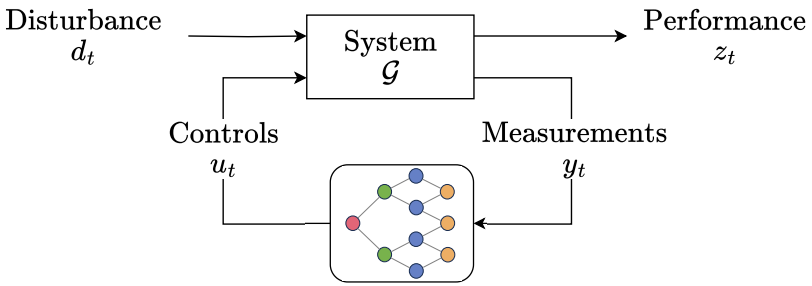
1. Motivation
2. Background: robust neural networks
3. Lipschitz-bounded policy networks
4. React to surprises with Youla-REN (Theory)
5. React to surprises with Youla-REN (Application)
6. Robust recurrent deep networks
7. Conclusions

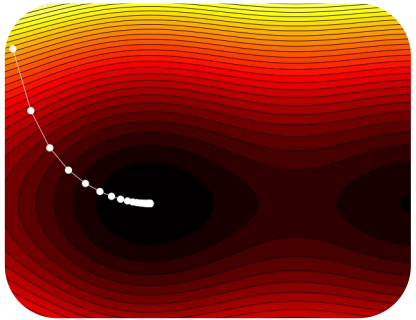
Stable-by-Design Neural Feedback Policies

Construct a new *policy class*:

1. Closed-loop robust stability by construction
2. Differentiable, scalable parametrisation
3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools

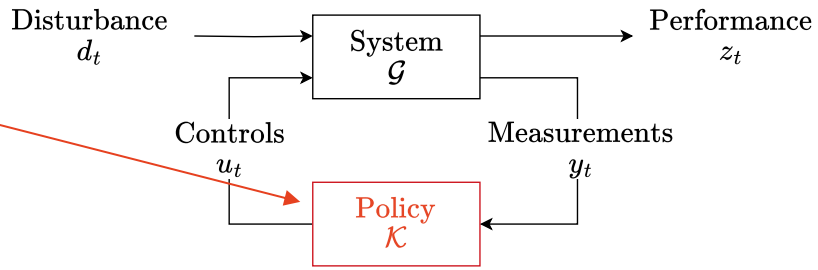




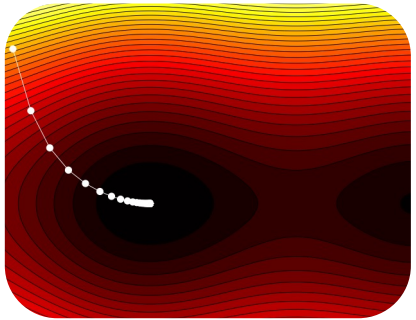
Stable-by-Design Neural Feedback Policies

Construct a new *policy class*:

1. Closed-loop robust stability by construction
2. Differentiable, scalable parametrisation
3. Expressive parametrisation

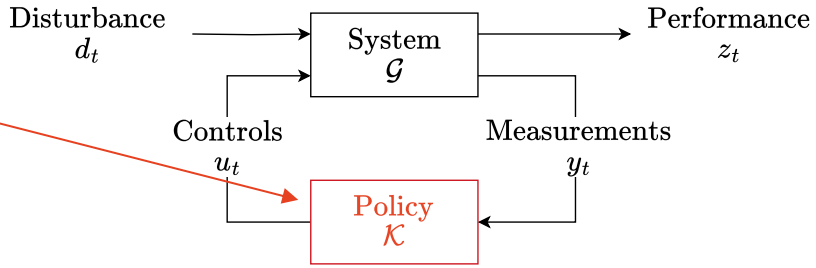


Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



Stable-by-Design Neural Feedback Policies

Construct a new *policy class*:

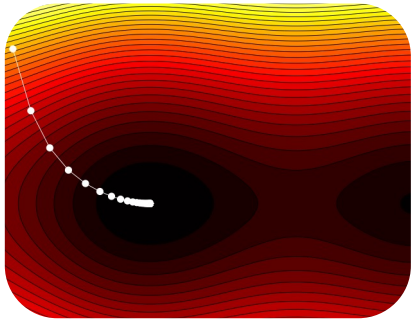


✅ 1. Closed-loop robust stability by construction

✅ 2. Differentiable, scalable parametrisation

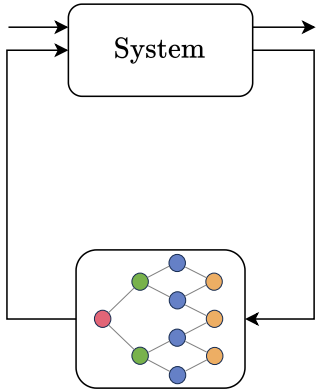
✅ 3. Expressive parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools



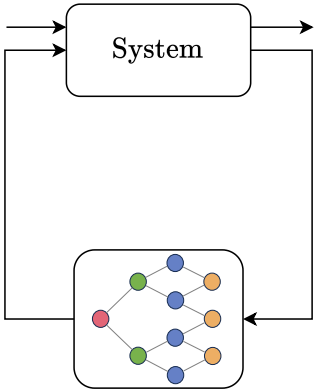
Policy Classes for Deep RL

Policy Classes for Deep RL

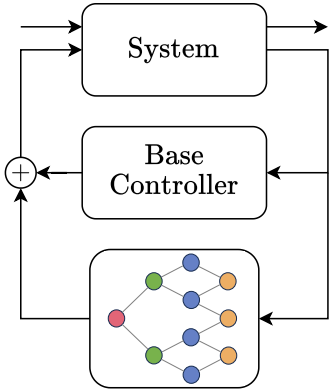


Vanilla RL

Policy Classes for Deep RL



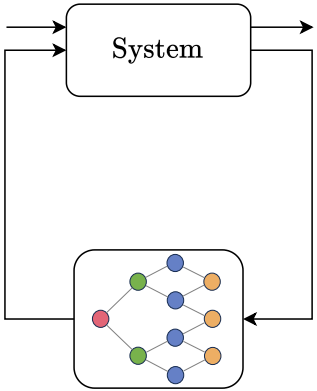
Vanilla RL



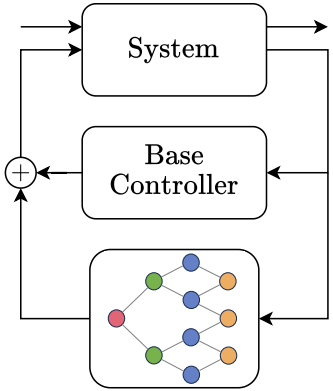
Residual RL[16]

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).

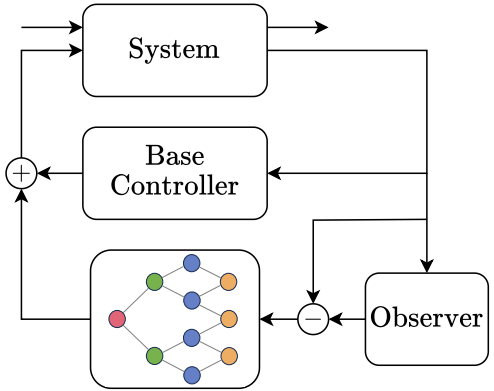
Policy Classes for Deep RL



Vanilla RL



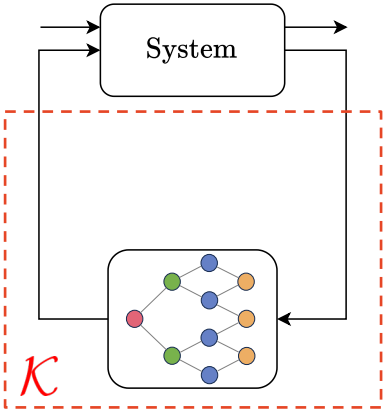
Residual RL[16]



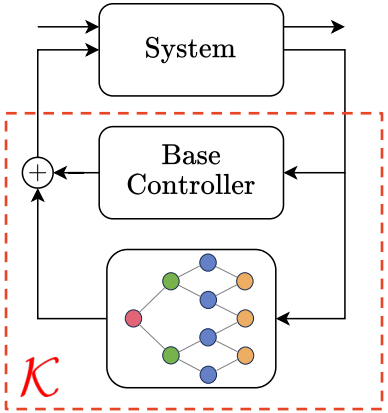
Youla-REN (ours)[17]

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

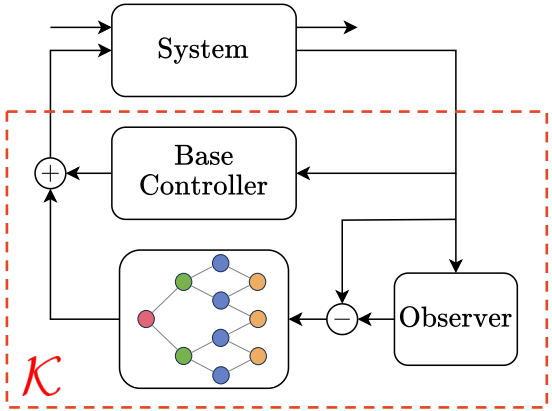
Policy Classes for Deep RL



Vanilla RL



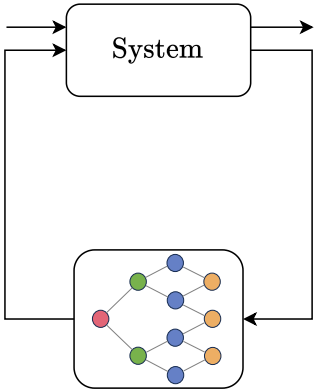
Residual RL[16]



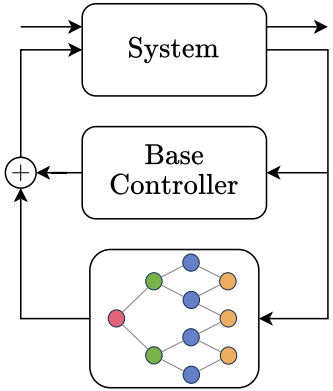
Youla-REN (ours)[17]

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

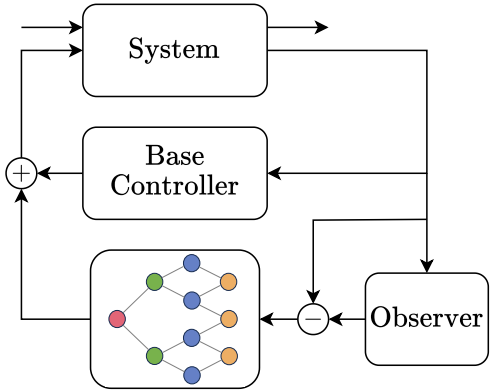
Policy Classes for Deep RL



Vanilla RL



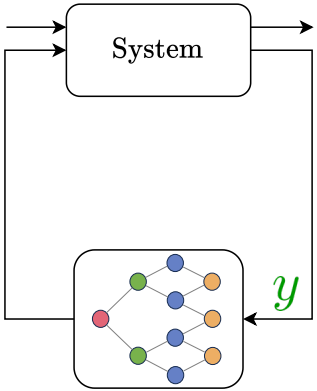
Residual RL[16]



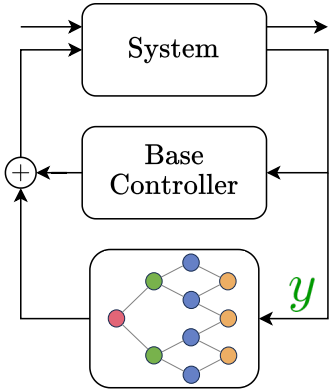
Youla-REN (ours)[17]

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

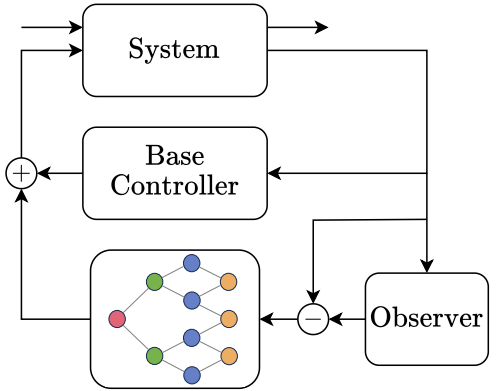
Policy Classes for Deep RL



Vanilla RL



Residual RL[16]

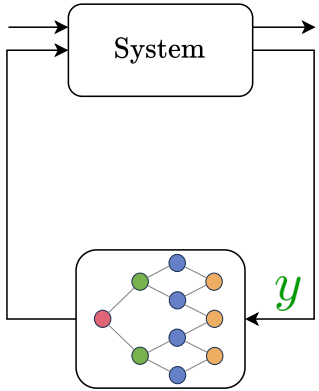


Youla-REN (ours)[17]

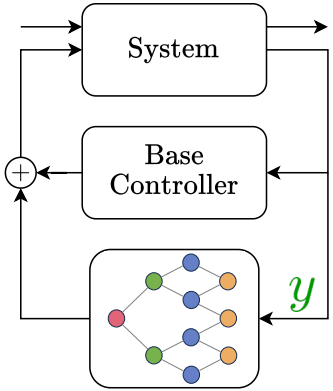
Network reacts to measurements y

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

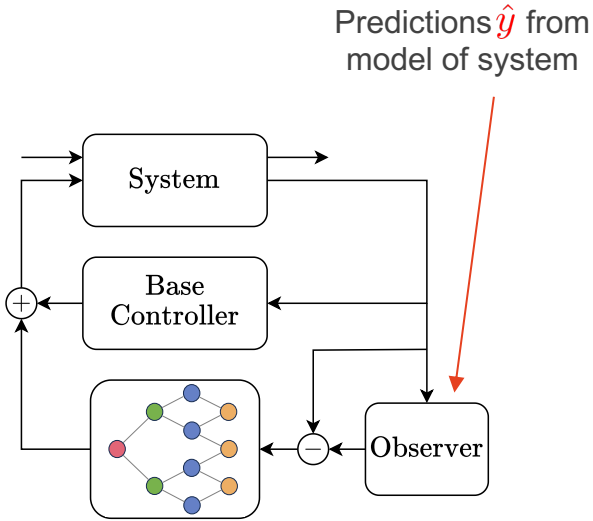
Policy Classes for Deep RL



Vanilla RL



Residual RL[16]

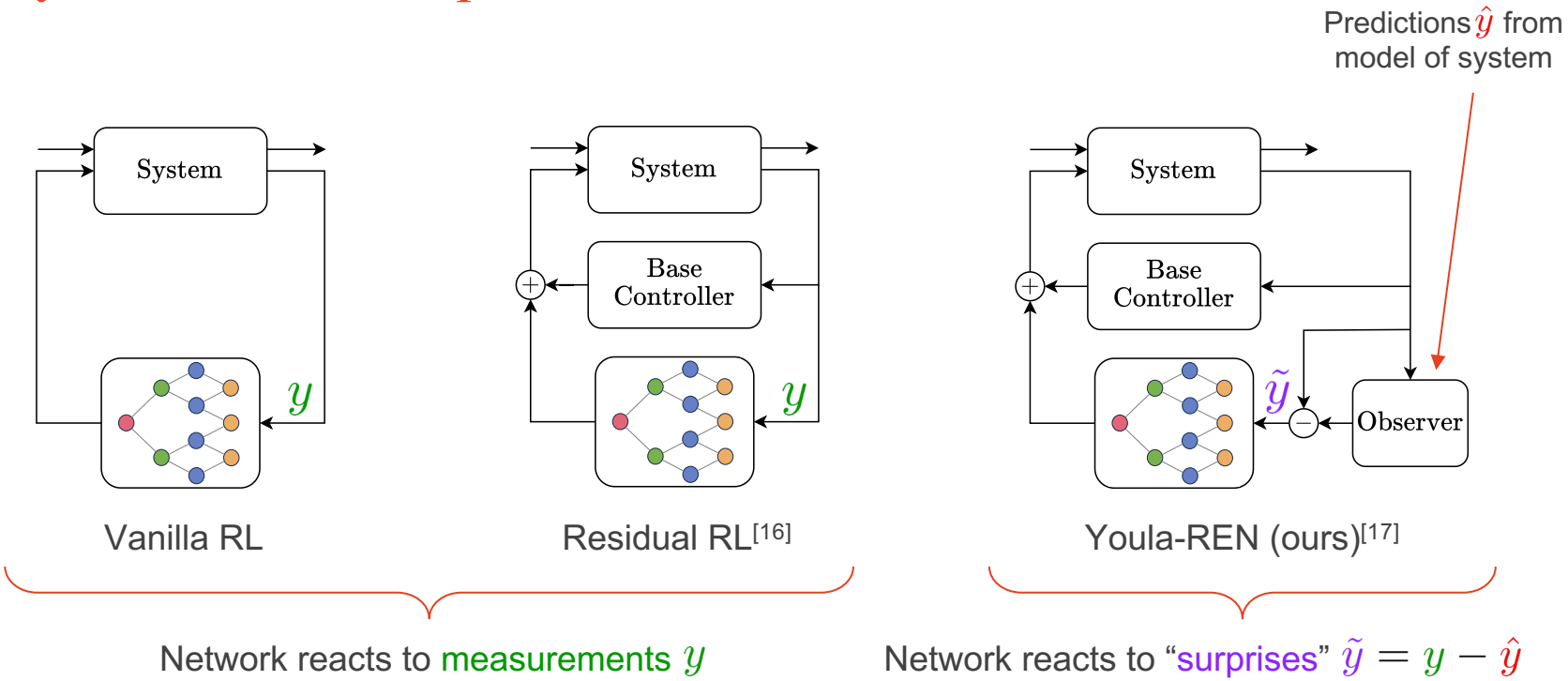


Youla-REN (ours)[17]

Network reacts to measurements y

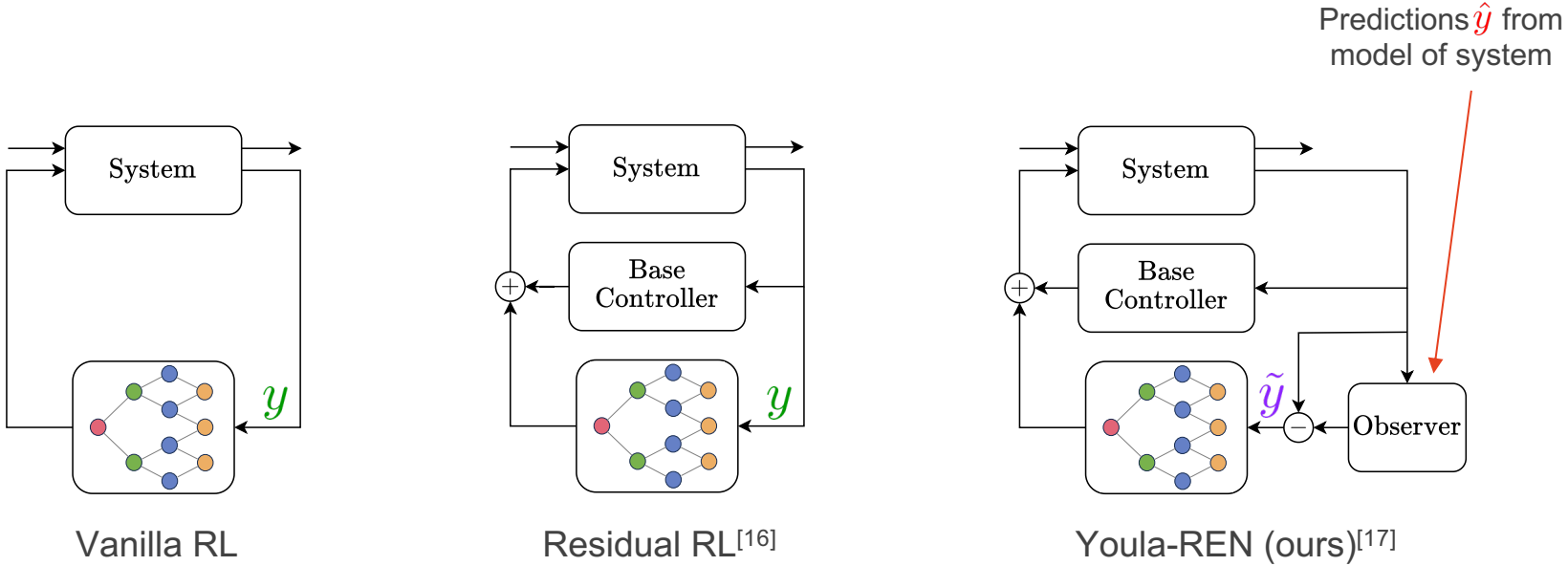
[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

Policy Classes for Deep RL



[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

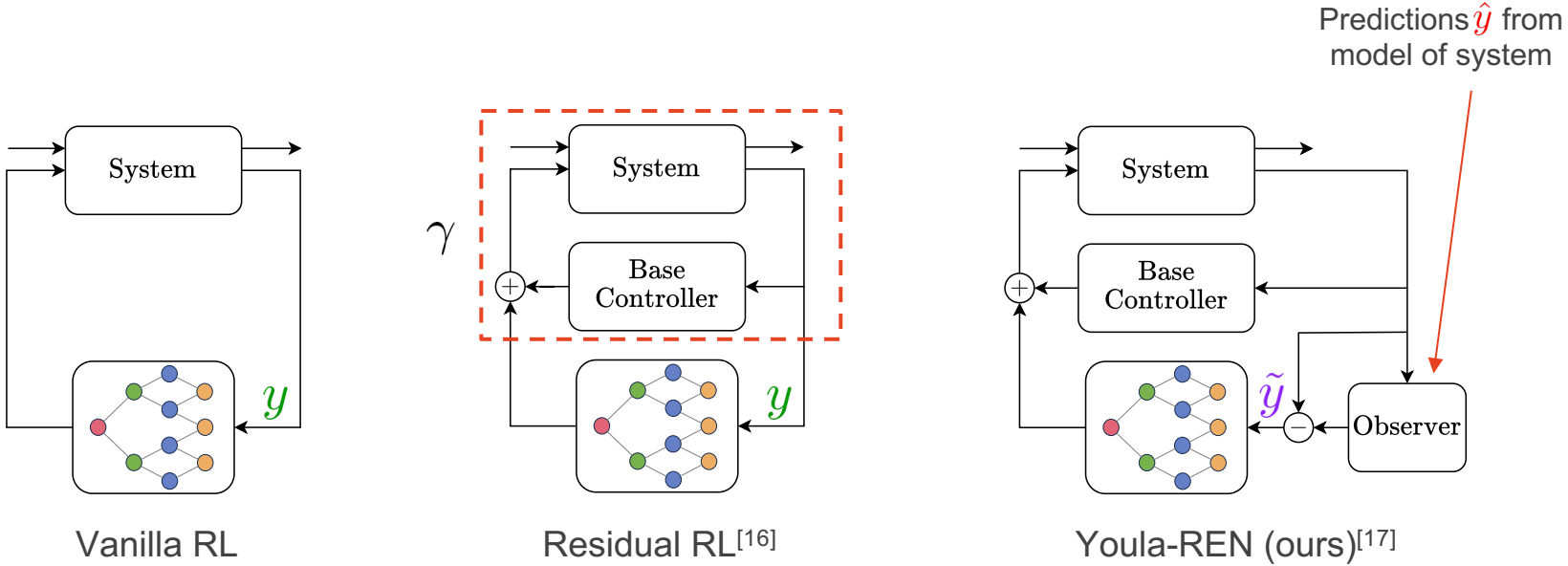
Policy Classes for Deep RL



- React to measurements y: closed-loop stability via small-gain theorem

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

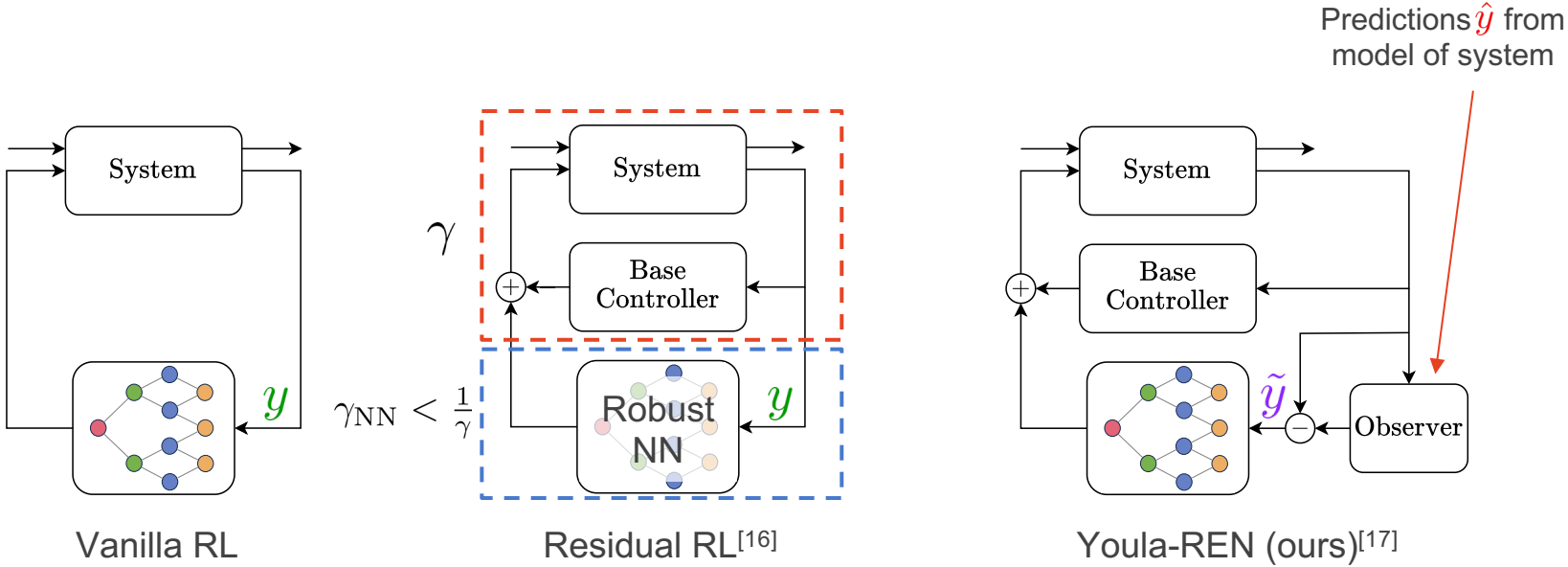
Policy Classes for Deep RL



- React to measurements y: closed-loop stability via small-gain theorem

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

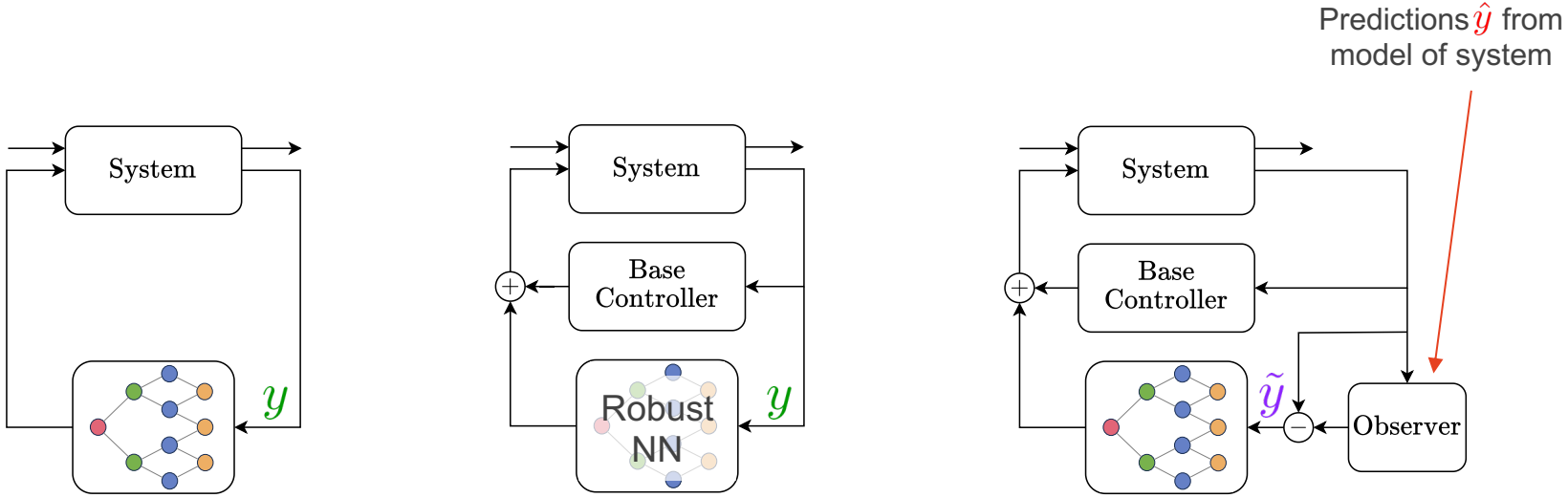
Policy Classes for Deep RL



- React to measurements y: closed-loop stability via small-gain theorem

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

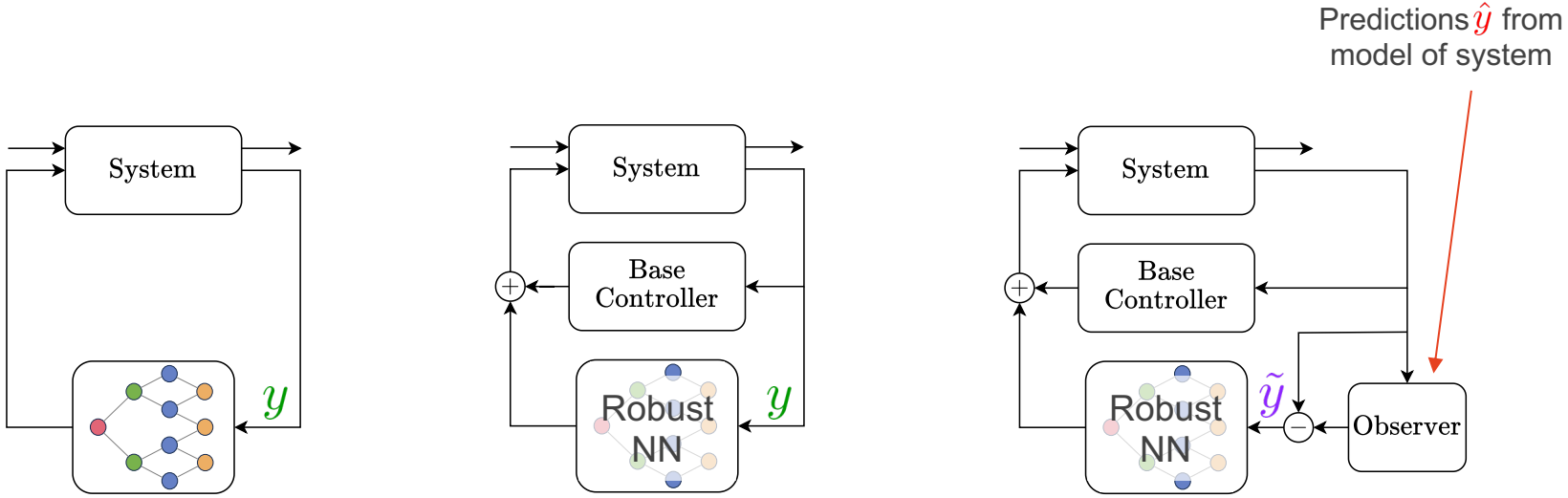
Policy Classes for Deep RL



- React to measurements y: closed-loop stability via small-gain theorem

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

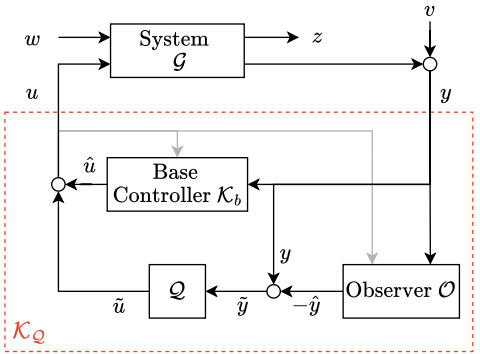
Policy Classes for Deep RL



- React to measurements y: closed-loop stability via small-gain theorem
- React to surprises $\tilde{y}$: closed-loop stability without gain restrictions (under assumptions)

[16] Johannik et al., Residual Reinforcement Learning for Robot Control, *ICRA* (2019).
[17] Barbara, Wang, & Manchester, React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

Youla Parametrisation Guarantees Closed-Loop Stability

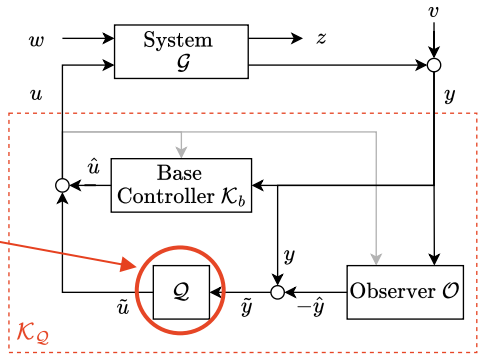


Youla Parametrisation Guarantees Closed-Loop Stability

"Youla Parameter" Q

Nonlinear (possibly dynamic) system.

Eg: neural network, or more general system



Youla Parametrisation Guarantees Closed-Loop Stability

Theorem. *Consider the LTI system*

$$\mathcal{G}: \begin{cases} x_{t+1} = Ax_t + Bu_t + w_t \\ y_t = Cx_t + v_t \end{cases}$$

together with an observer and base controller

$$\mathcal{O}, \mathcal{K}_b: \begin{cases} \hat{x}_{t+1} = A\hat{x}_t + Bu_t + L(y - C\hat{x}_t) \\ \hat{u}_t = -K\hat{x}_t \end{cases}$$

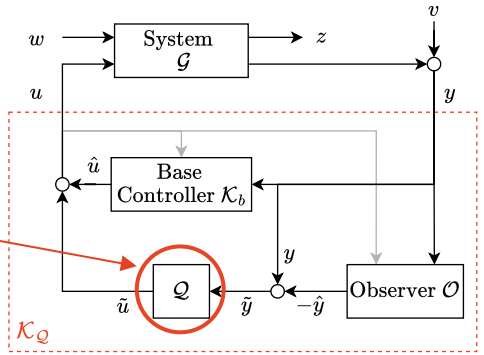
designed such that $(A - BK)$ and $(A - LC)$ are stable. Writing $d = [w; v]$ and $z = [x; u]$, then the following statement is true for the closed-loop system under the Youla controller $u = \hat{u} + \mathcal{Q}(y - C\hat{x})$:

A controller $u = \mathcal{K}(y)$ with a valid state-space representation can achieve a contracting and Lipschitz closed loop $d \mapsto z$ ***if and only if*** *it can be written in the form of the Youla controller* ***with a contracting and Lipschitz*** *$\mathcal{Q}$.*

"Youla Parameter" Q

Nonlinear (possibly dynamic) system.

Eg: neural network, or more general system



Youla Parametrisation Guarantees Closed-Loop Stability

Theorem. *Consider the LTI system*

$$\mathcal{G}: \begin{cases} x_{t+1} = Ax_t + Bu_t + w_t \\ y_t = Cx_t + v_t \end{cases}$$

together with an observer and base controller

$$\mathcal{O}, \mathcal{K}_b: \begin{cases} \hat{x}_{t+1} = A\hat{x}_t + Bu_t + L(y - C\hat{x}_t) \\ \hat{u}_t = -K\hat{x}_t \end{cases}$$

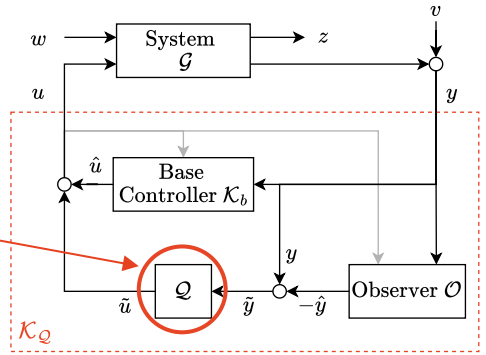
designed such that $(A - BK)$ *and* $(A - LC)$ *are stable. Writing* $d = [w; v]$ *and* $z = [x; u]$*, then the following statement is true for the closed-loop system under the Youla controller* $u = \hat{u} + \mathcal{Q}(y - C\hat{x})$*:*

A controller $u = \mathcal{K}(y)$ *with a valid state-space representation can achieve a contracting and Lipschitz closed loop* $d \mapsto z$ ***if and only if*** *it can be written in the form of the Youla controller* ***with a contracting and Lipschitz*** $\mathcal{Q}$*.*

"Youla Parameter" Q

Nonlinear (possibly dynamic) system.

Eg: neural network, or more general system



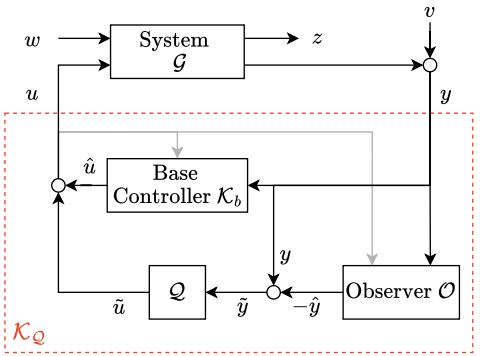
What it means:

- Stabilising controller $\iff$ Youla controller with a stable (contracting & Lipschitz) Q
- No gain restrictions on Q
- For linear[18] and nonlinear[19] controllers

I.e., the policy class $\mathcal{K}_\mathcal{Q}$ covers **all and only** stabilising controllers for LTI systems!

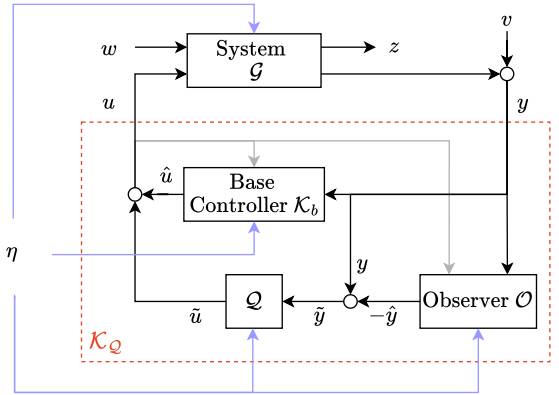
[18] Doyle, Matrix Interpolation Theory and Optimal Control, *PhD Thesis, Berkley* (1984).
[19] Wang, Barbara, Revay, & Manchester, Learning Over All Stabilizing Nonlinear Controllers for a Partially-Observed Linear System, *L-CSS* (2022).

Contracting & Lipschitz Closed Loops for Nonlinear Systems



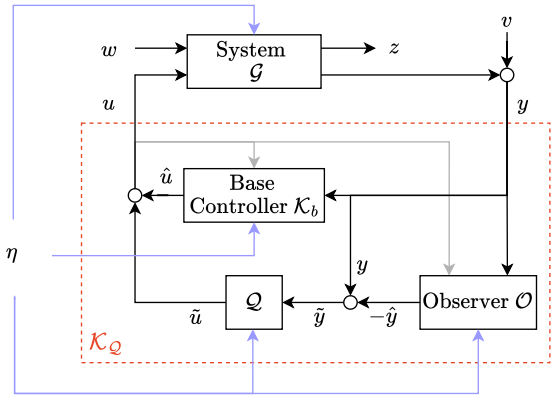
[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

Contracting & Lipschitz Closed Loops for Nonlinear Systems



[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

Contracting & Lipschitz Closed Loops for Nonlinear Systems



$$\mathcal{G}: \begin{cases} x_{t+1} = f(x_t, \eta_t, u_t) + w_t \\ y_t = h(x_t) + v_t \end{cases}$$

$$\mathcal{K}_b: \begin{cases} s_{t+1} = f_o(s_t, \eta_t, u_t, y_t) \\ \hat{u}_t = k(s_t, \eta_t, y_t) \end{cases}$$

$$\mathcal{O}: \begin{cases} \hat{x}_{t+1} = f_o(\hat{x}_t, \eta_t, u_t, y_t) \\ \hat{y}_t = h(\hat{x}_t) \end{cases}$$

$$\mathcal{Q}: \begin{cases} q_{t+1} = f_q(q_t, \eta_t, \tilde{y}_t) \\ \tilde{u}_t = h_q(q_t, \eta_t, \tilde{y}_t) \end{cases}$$

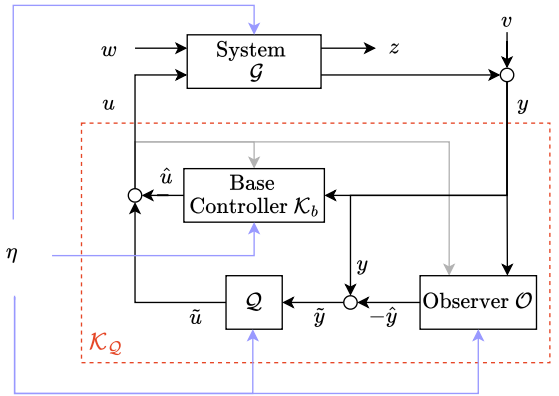
[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

Contracting & Lipschitz Closed Loops for Nonlinear Systems

Theorem. *Consider the partially-observed, nonlinear system $\mathcal{G}$ together with the base controller and observer $\mathcal{K}_b$ and $\mathcal{O}$ (respectively) which satisfy the following assumptions, where $d = [w; v]$ and $z = [x; u]$:*

- *$\mathcal{K}_b$ achieves a contracting & Lipschitz closed loop $(\eta, d, \tilde{u}) \mapsto z$ with $\mathcal{G}$.*
- *$\mathcal{O}$ is contracting and the map $(\eta, u, y) \mapsto \hat{x}$ is Lipschitz.*
- *Observer correctness: $f(x_t, \eta_t, u_t) = f_o(s_t, \eta_t, u_t, h(x_t))$.*
- *All systems have piecewise-diff'ble dynamics and Lipschitz output maps.*

Then, the following statements are true for the closed-loop system under the Youla controller $u = \hat{u} + \mathcal{Q}(y - \hat{y})$ ***with a contracting and Lipschitz*** *$\mathcal{Q}$:*



$$\mathcal{G}: \begin{cases} x_{t+1} = f(x_t, \eta_t, u_t) + w_t \\ y_t = h(x_t) + v_t \end{cases}$$

$$\mathcal{K}_b: \begin{cases} s_{t+1} = f_o(s_t, \eta_t, u_t, y_t) \\ \hat{u}_t = k(s_t, \eta_t, y_t) \end{cases}$$

$$\mathcal{O}: \begin{cases} \hat{x}_{t+1} = f_o(\hat{x}_t, \eta_t, u_t, y_t) \\ \hat{y}_t = h(\hat{x}_t) \end{cases}$$

$$\mathcal{Q}: \begin{cases} q_{t+1} = f_q(q_t, \eta_t, \tilde{y}_t) \\ \tilde{u}_t = h_q(q_t, \eta_t, \tilde{y}_t) \end{cases}$$

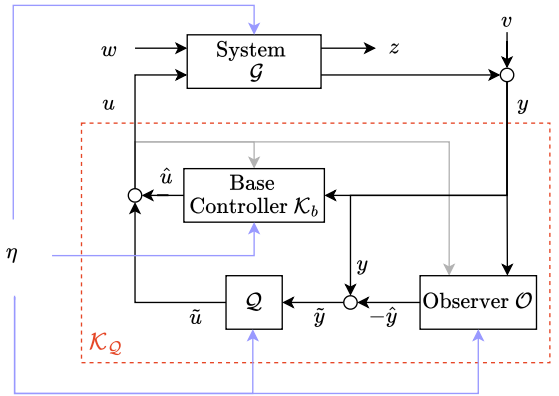
[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

Contracting & Lipschitz Closed Loops for Nonlinear Systems

Theorem. *Consider the partially-observed, nonlinear system $\mathcal{G}$ together with the base controller and observer $\mathcal{K}_b$ and $\mathcal{O}$ (respectively) which satisfy the following assumptions, where $d = [w; v]$ and $z = [x; u]$:*

- *$\mathcal{K}_b$ achieves a contracting & Lipschitz closed loop $(\eta, d, \tilde{u}) \mapsto z$ with $\mathcal{G}$.*
- *$\mathcal{O}$ is contracting and the map $(\eta, u, y) \mapsto \hat{x}$ is Lipschitz.*
- *Observer correctness: $f(x_t, \eta_t, u_t) = f_o(s_t, \eta_t, u_t, h(x_t))$.*
- *All systems have piecewise-diff'ble dynamics and Lipschitz output maps.*

Then, the following statements are true for the closed-loop system under the Youla controller $u = \hat{u} + \mathcal{Q}(y - \hat{y})$ ***with a contracting and Lipschitz*** *$\mathcal{Q}$:*



$$\mathcal{G}: \begin{cases} x_{t+1} = f(x_t, \eta_t, u_t) + w_t \\ y_t = h(x_t) + v_t \end{cases}$$

$$\mathcal{K}_b: \begin{cases} s_{t+1} = f_o(s_t, \eta_t, u_t, y_t) \\ \hat{u}_t = k(s_t, \eta_t, y_t) \end{cases}$$

$$\mathcal{O}: \begin{cases} \hat{x}_{t+1} = f_o(\hat{x}_t, \eta_t, u_t, y_t) \\ \hat{y}_t = h(\hat{x}_t) \end{cases}$$

$$\mathcal{Q}: \begin{cases} q_{t+1} = f_q(q_t, \eta_t, \tilde{y}_t) \\ \tilde{u}_t = h_q(q_t, \eta_t, \tilde{y}_t) \end{cases}$$

[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

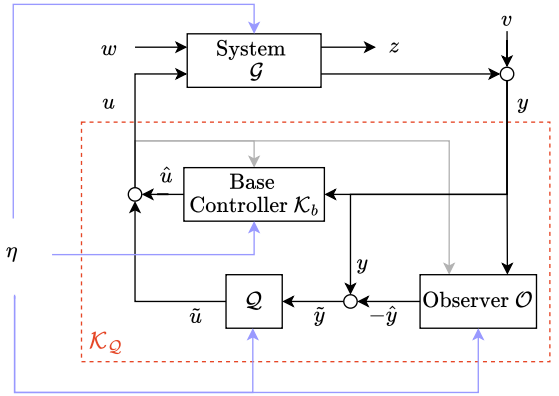
Contracting & Lipschitz Closed Loops for Nonlinear Systems

Theorem. *Consider the partially-observed, nonlinear system $\mathcal{G}$ together with the base controller and observer $\mathcal{K}_b$ and $\mathcal{O}$ (respectively) which satisfy the following assumptions, where $d = [w; v]$ and $z = [x; u]$:*

- *$\mathcal{K}_b$ achieves a contracting & Lipschitz closed loop $(\eta, d, \tilde{u}) \mapsto z$ with $\mathcal{G}$.*
- *$\mathcal{O}$ is contracting and the map $(\eta, u, y) \mapsto \hat{x}$ is Lipschitz.*
- *Observer correctness: $f(x_t, \eta_t, u_t) = f_o(s_t, \eta_t, u_t, h(x_t))$.*
- *All systems have piecewise-diff'ble dynamics and Lipschitz output maps.*

Then, the following statements are true for the closed-loop system under the Youla controller $u = \hat{u} + \mathcal{Q}(y - \hat{y})$ ***with a contracting and Lipschitz*** *$\mathcal{Q}$:*

1. *With zero disturbances (i.e., $d \equiv 0$, $x_0 = \hat{x}_0$), the closed-loop system is contracting and $\eta \mapsto z$ is Lipschitz.*



$$\mathcal{G}: \begin{cases} x_{t+1} = f(x_t, \eta_t, u_t) + w_t \\ y_t = h(x_t) + v_t \end{cases}$$

$$\mathcal{K}_b: \begin{cases} s_{t+1} = f_o(s_t, \eta_t, u_t, y_t) \\ \hat{u}_t = k(s_t, \eta_t, y_t) \end{cases}$$

$$\mathcal{O}: \begin{cases} \hat{x}_{t+1} = f_o(\hat{x}_t, \eta_t, u_t, y_t) \\ \hat{y}_t = h(\hat{x}_t) \end{cases}$$

$$\mathcal{Q}: \begin{cases} q_{t+1} = f_q(q_t, \eta_t, \tilde{y}_t) \\ \tilde{u}_t = h_q(q_t, \eta_t, \tilde{y}_t) \end{cases}$$

[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

Contracting & Lipschitz Closed Loops for Nonlinear Systems

Theorem. *Consider the partially-observed, nonlinear system $\mathcal{G}$ together with the base controller and observer $\mathcal{K}_b$ and $\mathcal{O}$ (respectively) which satisfy the following assumptions, where $d = [w; v]$ and $z = [x; u]$:*

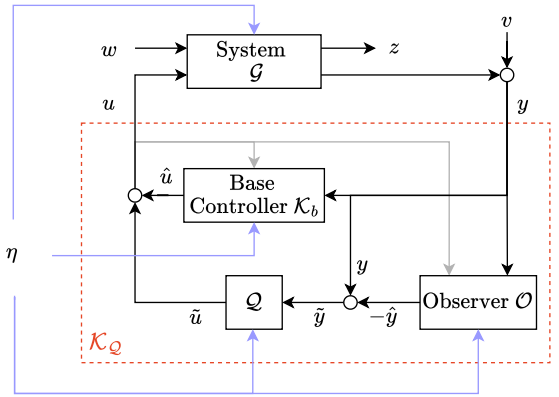
- *$\mathcal{K}_b$ achieves a contracting & Lipschitz closed loop $(\eta, d, \tilde{u}) \mapsto z$ with $\mathcal{G}$.*
- *$\mathcal{O}$ is contracting and the map $(\eta, u, y) \mapsto \hat{x}$ is Lipschitz.*
- *Observer correctness: $f(x_t, \eta_t, u_t) = f_o(s_t, \eta_t, u_t, h(x_t))$.*
- *All systems have piecewise-diff'ble dynamics and Lipschitz output maps.*

Then, the following statements are true for the closed-loop system under the Youla controller $u = \hat{u} + \mathcal{Q}(y - \hat{y})$ ***with a contracting and Lipschitz*** *$\mathcal{Q}$:*

1. *With zero disturbances (i.e., $d \equiv 0$, $x_0 = \hat{x}_0$), the closed-loop system is contracting and $\eta \mapsto z$ is Lipschitz.*
2. *Let * denote any zero-disturbance closed-loop trajectory. Then with non-zero disturbances (i.e., $d \not\equiv 0$, $x_0 \neq \hat{x}_0$), the closed-loop system satisfies*

$$\|z - z^\star\|_T \leq \gamma(\|\eta - \eta^\star\|_T + \|d\|_T) + \kappa(\bar{x}_0, \bar{x}_0^\star)$$

for all $T \in \mathbb{N}$, $\bar{x}_0, \eta$, where $\gamma \in \mathbb{R}^+$, $\kappa(\bar{x}_0, \bar{x}_0^\star) \geq 0$, $\kappa(\bar{x}_0^\star, \bar{x}_0^\star) = 0$. We call this property **d-tube contracting and Lipschitz**.



$$\mathcal{G}: \begin{cases} x_{t+1} = f(x_t, \eta_t, u_t) + w_t \\ y_t = h(x_t) + v_t \end{cases}$$

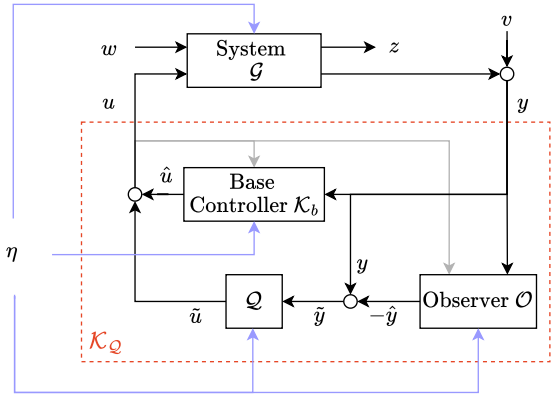
$$\mathcal{K}_b: \begin{cases} s_{t+1} = f_o(s_t, \eta_t, u_t, y_t) \\ \hat{u}_t = k(s_t, \eta_t, y_t) \end{cases}$$

$$\mathcal{O}: \begin{cases} \hat{x}_{t+1} = f_o(\hat{x}_t, \eta_t, u_t, y_t) \\ \hat{y}_t = h(\hat{x}_t) \end{cases}$$

$$\mathcal{Q}: \begin{cases} q_{t+1} = f_q(q_t, \eta_t, \tilde{y}_t) \\ \tilde{u}_t = h_q(q_t, \eta_t, \tilde{y}_t) \end{cases}$$

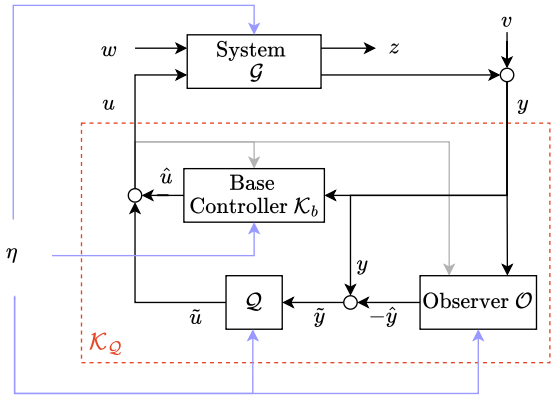
[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

d-Tube: What it means



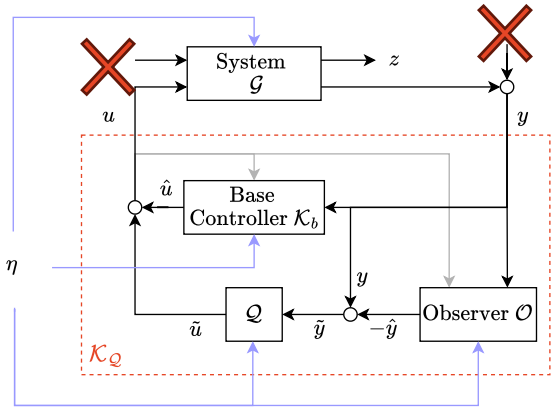
d-Tube: What it means

- Known inputs: the closed-loop system of $\mathcal{K}_{\mathcal{Q}}$ and $\mathcal{G}$ is contracting & Lipschitz



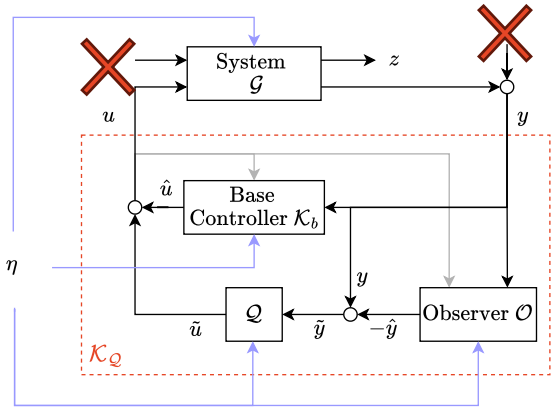
d-Tube: What it means

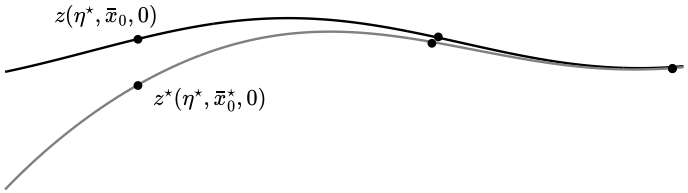
- Known inputs: the closed-loop system of $\mathcal{K}_\mathcal{Q}$ and $\mathcal{G}$ is contracting & Lipschitz



d-Tube: What it means

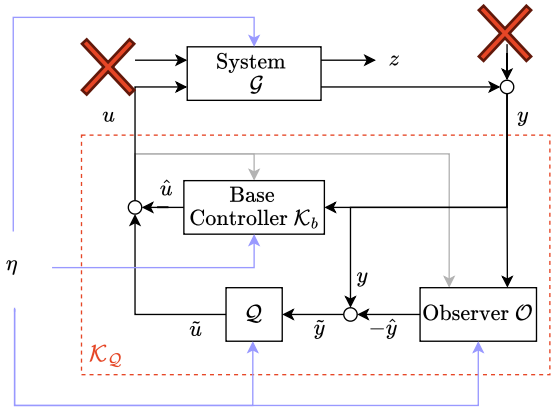
- Known inputs: the closed-loop system of $\mathcal{K}_\mathcal{Q}$ and $\mathcal{G}$ is contracting & Lipschitz

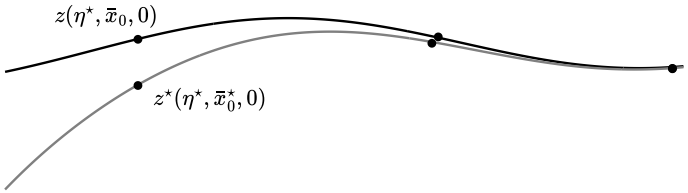




d-Tube: What it means

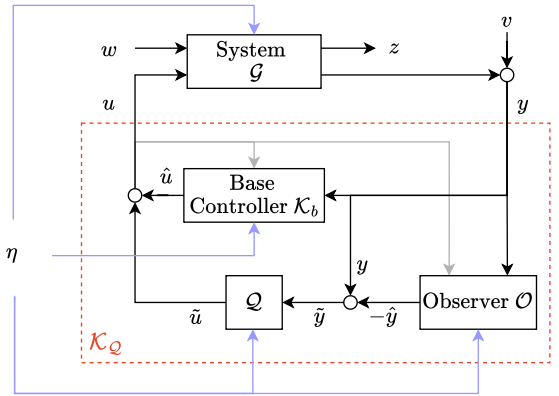
- Known inputs: the closed-loop system of $\mathcal{K}_\mathcal{Q}$ and $\mathcal{G}$ is contracting & Lipschitz
- Unknown inputs: trajectories converge to a tube around the disturbance-free trajectories.

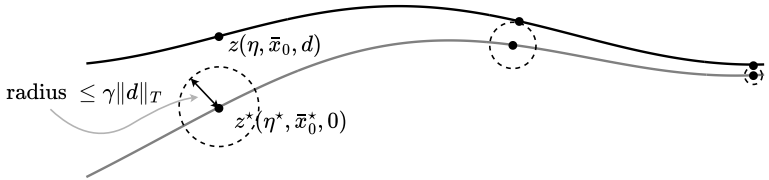




d-Tube: What it means

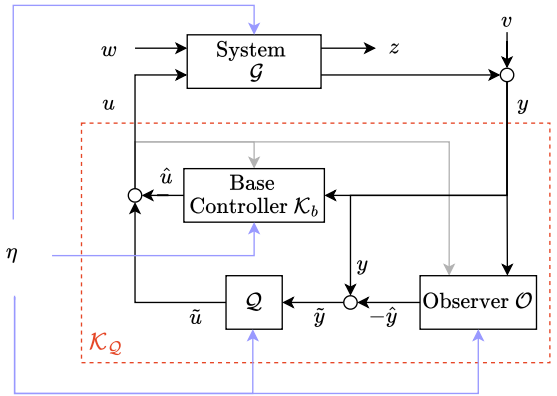
- Known inputs: the closed-loop system of $\mathcal{K}_\mathcal{Q}$ and $\mathcal{G}$ is contracting & Lipschitz
- Unknown inputs: trajectories converge to a tube around the disturbance-free trajectories.

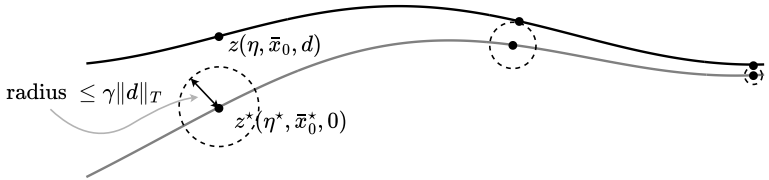




d-Tube: What it means

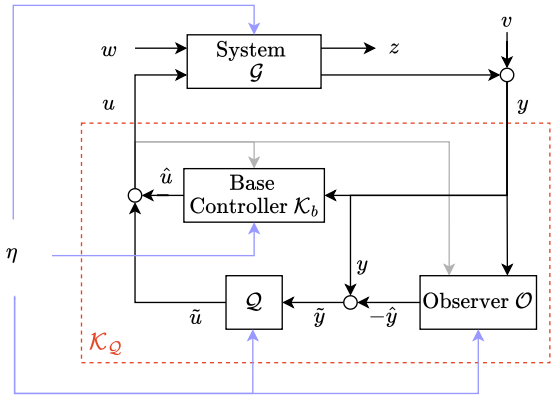
- Known inputs: the closed-loop system of $\mathcal{K}_\mathcal{Q}$ and $\mathcal{G}$ is contracting & Lipschitz
- Unknown inputs: trajectories converge to a tube around the disturbance-free trajectories.
- Is this useful?
  - Yes – practical stability
  - Smooth response to known inputs
  - Bounded response to bounded unknown inputs

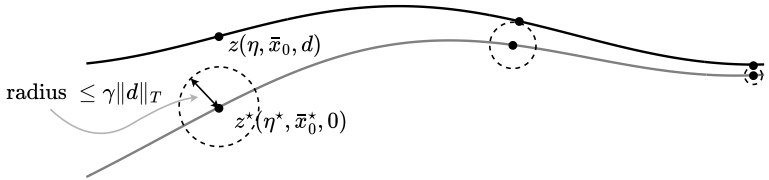




d-Tube: What it means

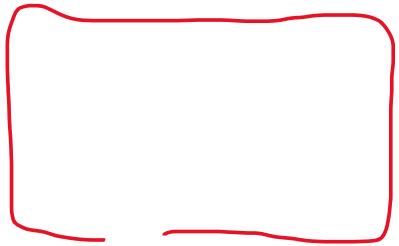
- Known inputs: the closed-loop system of $\mathcal{K}_\mathcal{Q}$ and $\mathcal{G}$ is contracting & Lipschitz
- Unknown inputs: trajectories converge to a tube around the disturbance-free trajectories.
- Is this useful?
  - Yes – practical stability
  - Smooth response to known inputs
  - Bounded response to bounded unknown inputs
- Converse results: see thesis or paper[17] ☺





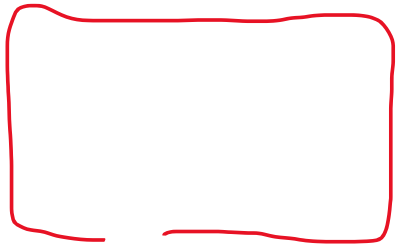
[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

d-Tube: Why it’s different

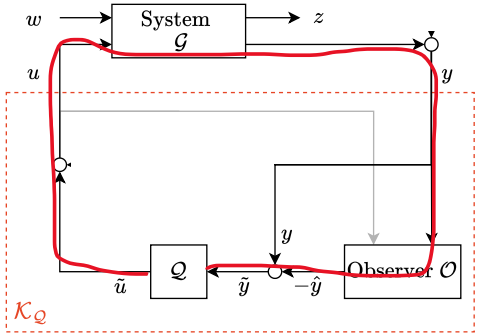


d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz



d-Tube: Why it's different



- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

- Example[20]:

$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

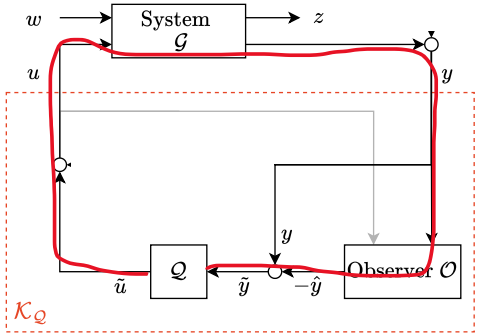
d-Tube: Why it's different

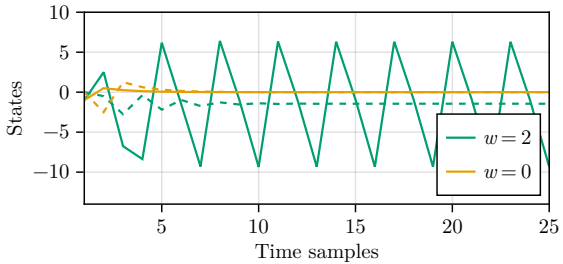
- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

- Example[20]:

$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$





[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

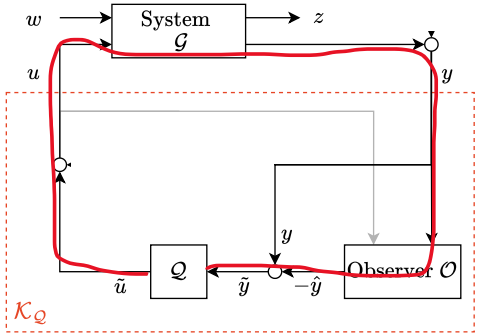
- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

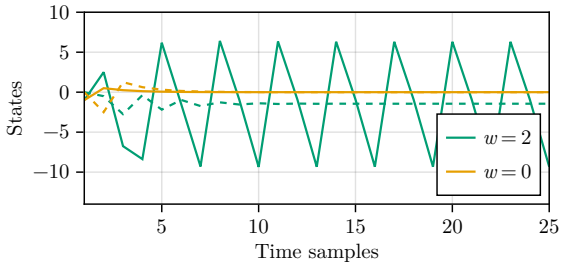
- Example[20]:

$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- What's happening?





[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

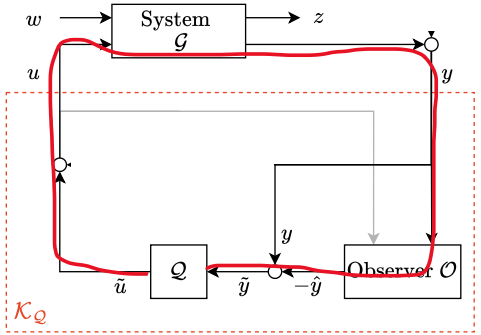
- Example[20]:

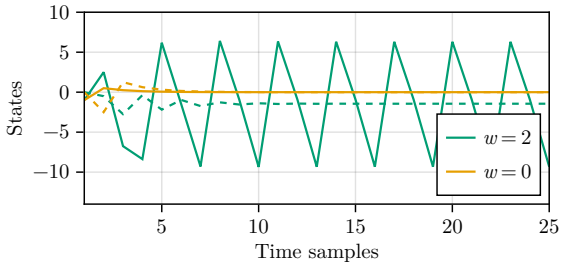
$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- What's happening?

$$\tilde{y}_t = y_t - \hat{y}_t = \tilde{x}_t$$
$$\tilde{x}_{t+1} = 0.5|\tilde{x}_t + \hat{x}_t| - 0.5|\hat{x}_t| + w_t$$





[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

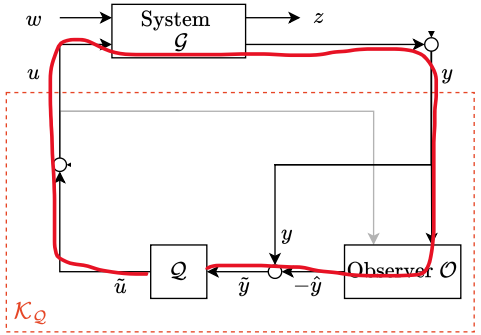
- Example[20]:

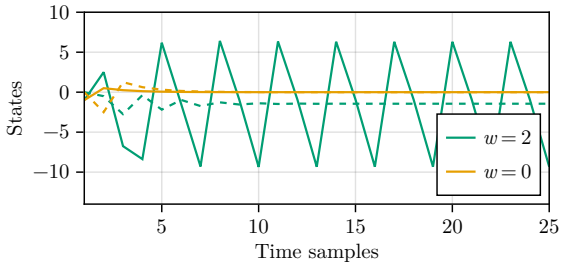
$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- What's happening?

$$\tilde{y}_t = y_t - \hat{y}_t = \tilde{x}_t$$
$$\tilde{x}_{t+1} = 0.5|\tilde{x}_t + \hat{x}_t| - 0.5|\hat{x}_t| + w_t$$





- $\tilde{y}$ depends on u, which depends on $\tilde{y}$, etc…
- High-gain Q can de-stabilize the feedback loop

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

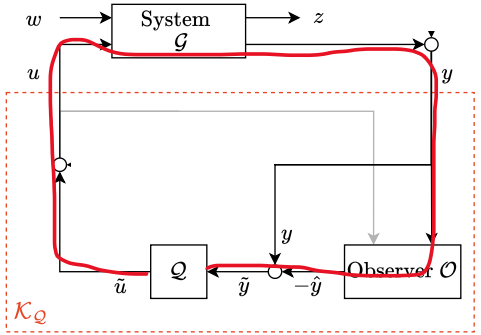
- Example[20]:

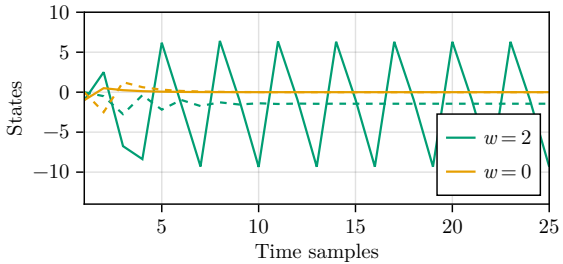
$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- What's happening?

$$\tilde{y}_t = y_t - \hat{y}_t = \tilde{x}_t$$
$$\tilde{x}_{t+1} = 0.5|\tilde{x}_t + \hat{x}_t| - 0.5|\hat{x}_t| + w_t$$





- $\tilde{y}$ depends on u, which depends on $\tilde{y}$, etc…
- High-gain Q can de-stabilize the feedback loop

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

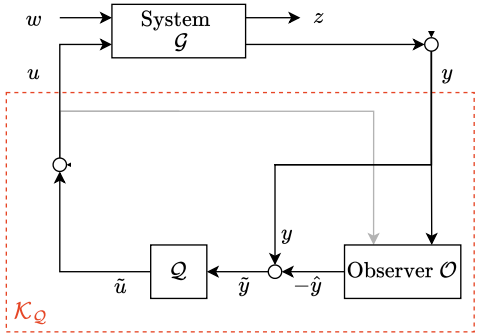
- Example[20]:

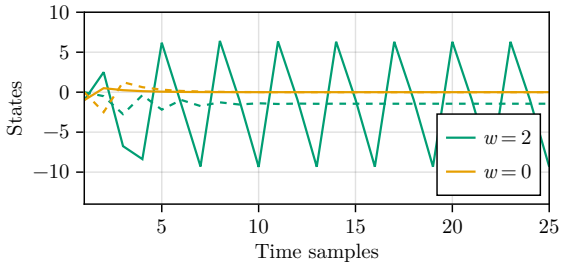
$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- What's happening?

$$\tilde{y}_t = y_t - \hat{y}_t = \tilde{x}_t$$
$$\tilde{x}_{t+1} = 0.5|\tilde{x}_t + \hat{x}_t| - 0.5|\hat{x}_t| + w_t$$





- Linear case: $\hat{x}$ terms cancel out

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

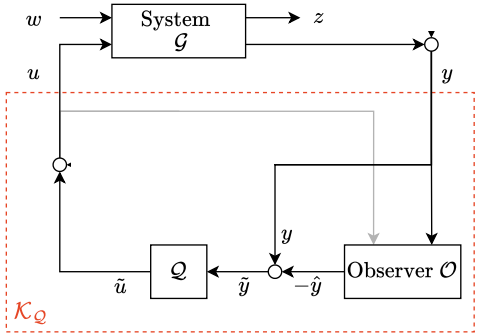
- Example[20]:

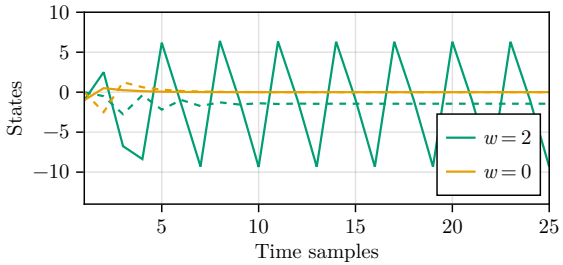
$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- What's happening?

$$\tilde{y}_t = y_t - \hat{y}_t = \tilde{x}_t$$
$$\tilde{x}_{t+1} = 0.5|\tilde{x}_t + \hat{x}_t| - 0.5|\hat{x}_t| + w_t$$





- Linear case: $\hat{x}$ terms cancel out

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

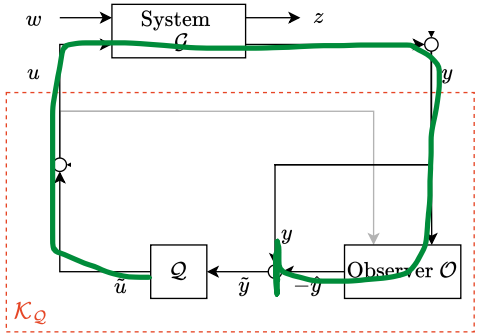
- Example[20]:

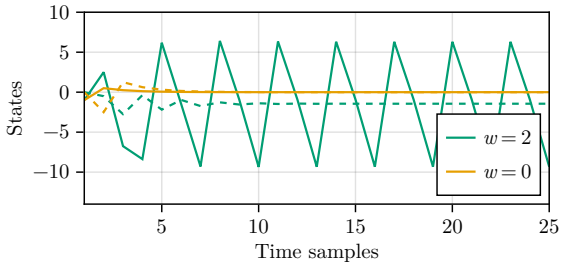
$$\mathcal{G}: \quad x_{t+1} = 0.5x_t + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5\hat{x}_t + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- What's happening?

$$\tilde{y}_t = y_t - \hat{y}_t = \tilde{x}_t$$
$$\tilde{x}_{t+1} = 0.5\tilde{x}_t + w_t$$





- Linear case: $\hat{x}$ terms cancel out

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

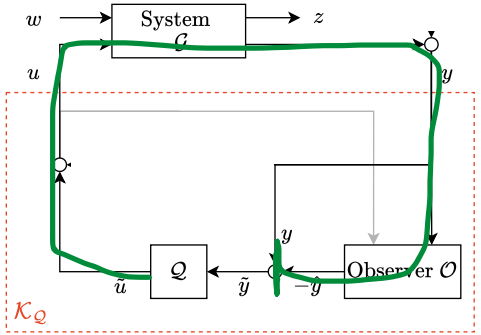
d-Tube: Why it's different

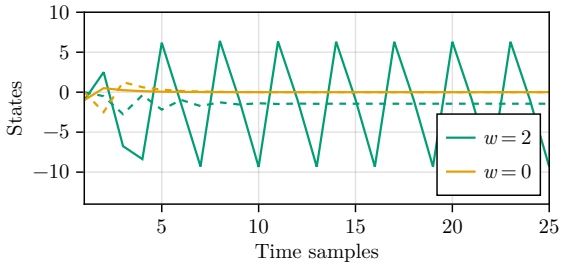
- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

- Example[20]:

$$\mathcal{G}: \quad x_{t+1} = 0.5x_t + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5\hat{x}_t + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$





- What's happening?

$$\tilde{y}_t = y_t - \hat{y}_t = \tilde{x}_t$$
$$\tilde{x}_{t+1} = 0.5\tilde{x}_t + w_t$$

Surprises have no u dependence!

- Linear case: $\hat{x}$ terms cancel out

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

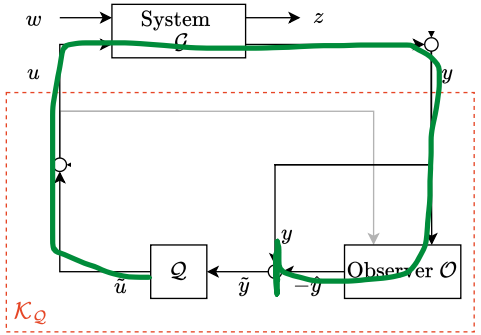
d-Tube: Why it's different

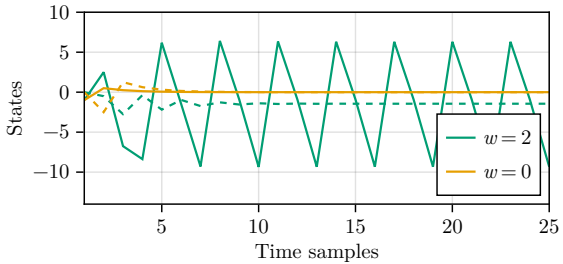
- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

- Example[20]:

$$\mathcal{G}: \quad x_{t+1} = 0.5x_t + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5\hat{x}_t + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$





- What's happening?

$$\tilde{y}_t = y_t - \hat{y}_t = \tilde{x}_t$$
$$\tilde{x}_{t+1} = 0.5\tilde{x}_t + w_t$$

Surprises have no u dependence!

- Linear case: $\hat{x}$ terms cancel out

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

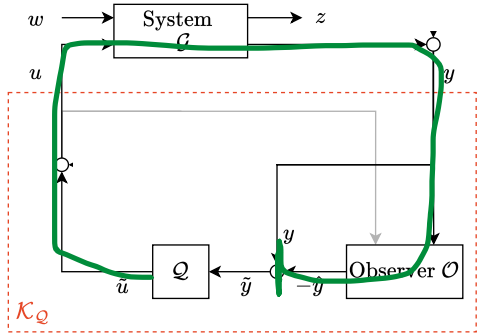
d-Tube: Why it's different

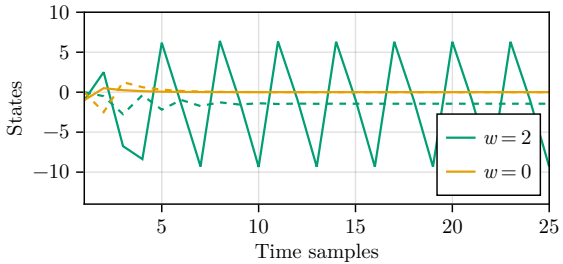
- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

- Example[20]:

$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$





[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

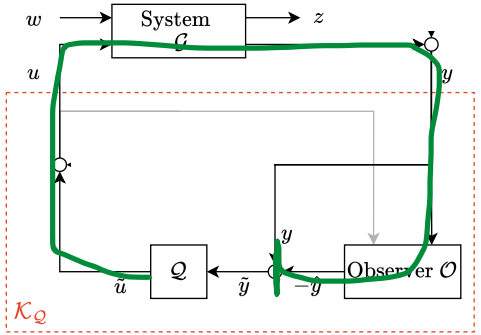
- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

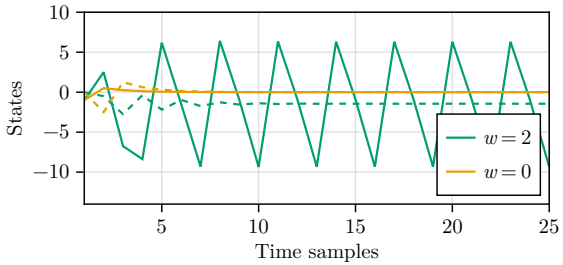
- Example[20]:

$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- Why's it still bounded for nonlinear?





[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

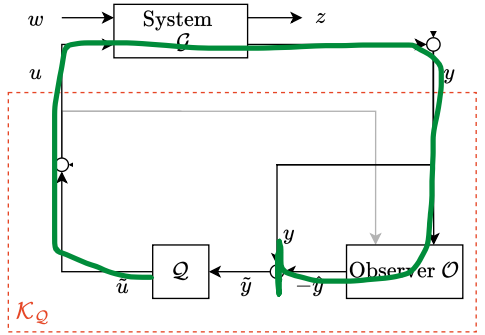
- Example[20]:

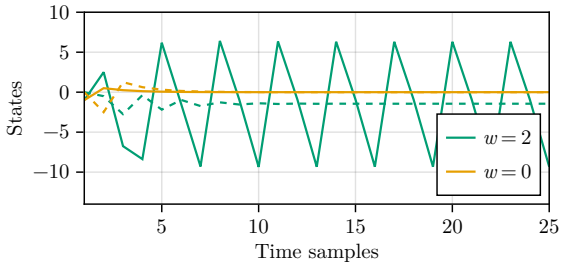
$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- Why's it still bounded for nonlinear?

- Can show: $\|\tilde{y}\|_T \leq \gamma_1 \|w\|_T + \gamma_2 |\tilde{x}_0|$





[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

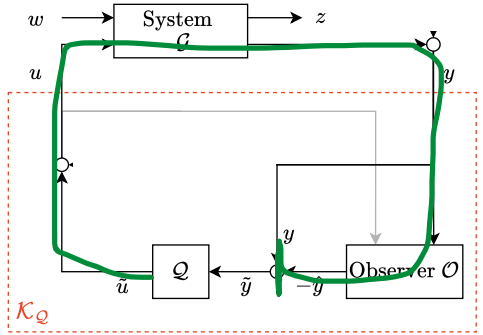
- Example[20]:

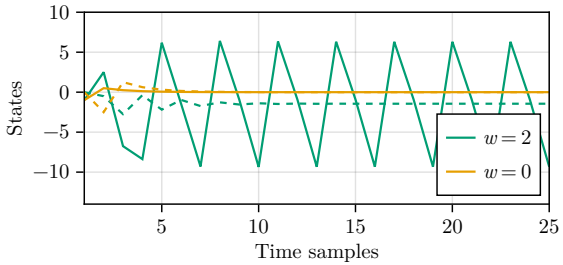
$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- Why's it still bounded for nonlinear?

- Can show: $\|\tilde{y}\|_T \leq \gamma_1 \|w\|_T + \gamma_2 |\tilde{x}_0|$





Bound on surprises has no u dependence!

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

d-Tube: Why it's different

- Linear systems: Youla $\Longrightarrow$ contracting & Lipschitz
- Nonlinear systems: Youla $\not\Longrightarrow$ contracting & Lipschitz

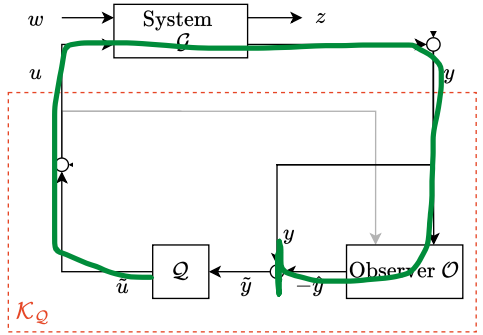
- Example[20]:

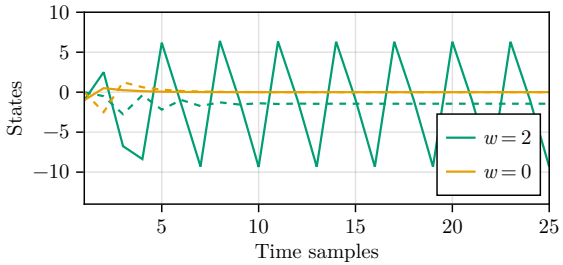
$$\mathcal{G}: \quad x_{t+1} = 0.5|x_t| + u_t + w_t, \quad y_t = x_t$$
$$\mathcal{O}: \quad \hat{x}_{t+1} = 0.5|\hat{x}_t| + u_t, \quad \hat{y}_t = \hat{x}_t$$

$$u = \mathcal{Q}(\tilde{y}) := \min(2.5 - 5\tilde{y}, 0)$$

- Why's it still bounded for nonlinear?

- Can show: $\|\tilde{y}\|_T \leq \gamma_1 \|w\|_T + \gamma_2 |\tilde{x}_0|$





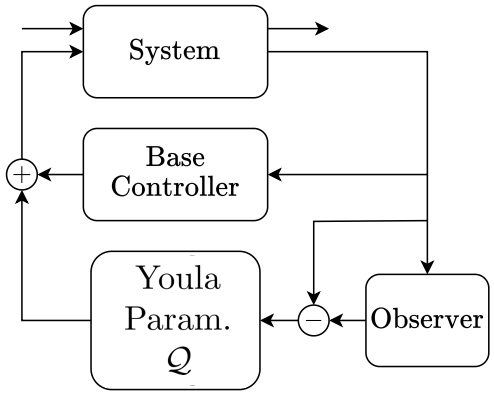
Bound on surprises has no u dependence!

[20] Inspired by a conversation with Prof. A. Megretski one Friday afternoon in Boston (2023).

Outline

1. Motivation
2. Background: robust neural networks
3. Lipschitz-bounded policy networks
4. React to surprises with Youla-REN (Theory)
5. React to surprises with Youla-REN (Application)
6. Robust recurrent deep networks
7. Conclusions

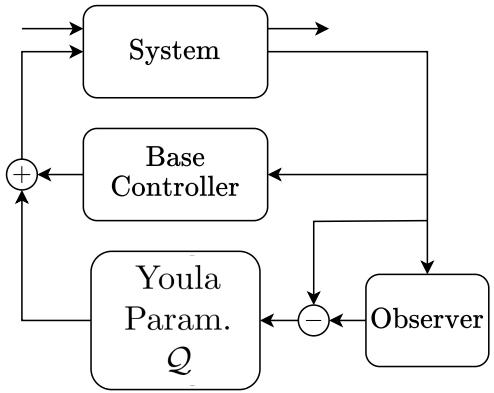
The Youla-REN



[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

The Youla-REN

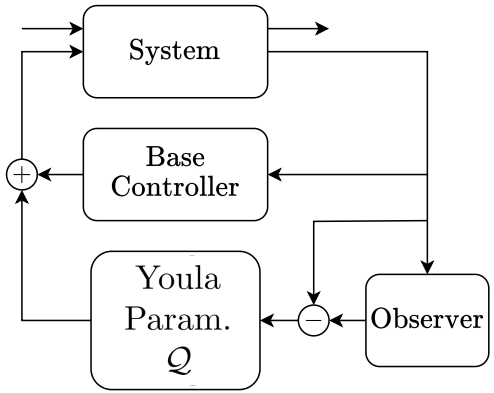
- Theory says: closed-loop stability guaranteed if we have contracting & Lipschitz Q



[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

The Youla-REN

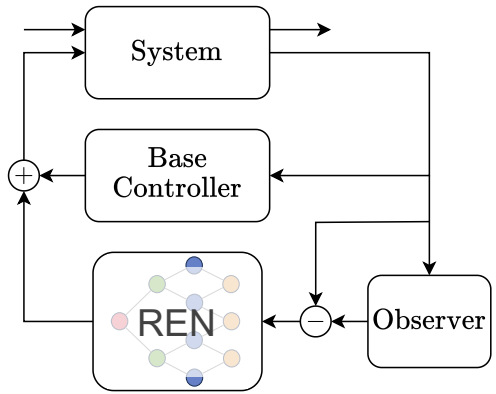
- Theory says: closed-loop stability guaranteed if we have contracting & Lipschitz Q
- Implementation: Recurrent Equlibrium Network (REN)

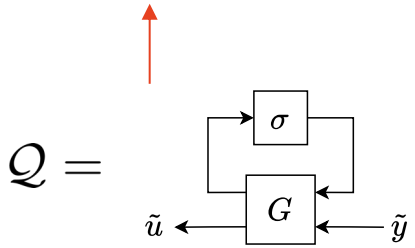


[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

The Youla-REN

- Theory says: closed-loop stability guaranteed if we have contracting & Lipschitz Q
- Implementation: Recurrent Equlibrium Network (REN)

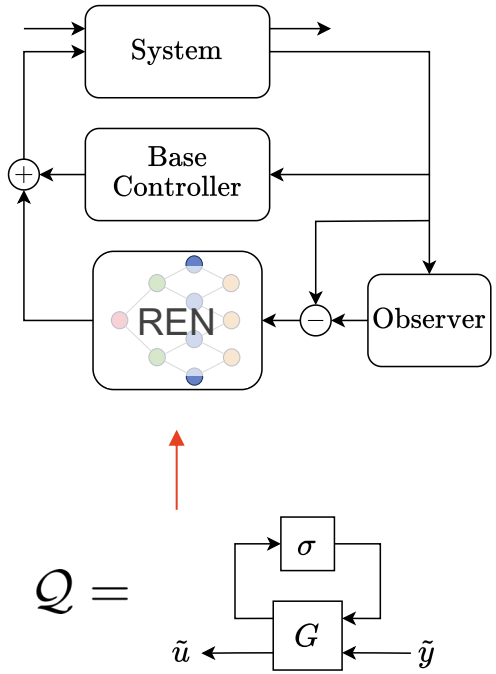




[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

The Youla-REN

- Theory says: closed-loop stability guaranteed if we have contracting & Lipschitz Q
- Implementation: Recurrent Equlibrium Network (REN)
  - Contracting & Lipschitz
  - Universal approximator
  - Direct parametrisation

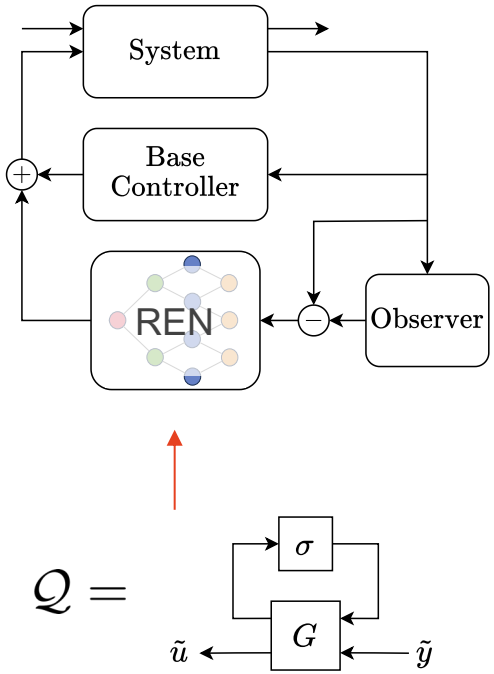


[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

The Youla-REN

- Theory says: closed-loop stability guaranteed if we have contracting & Lipschitz Q
- Implementation: Recurrent Equlibrium Network (REN)
  - Contracting & Lipschitz
  - Universal approximator
  - Direct parametrisation

Directly learn controllers with robust stability guarantees using standard ML/deep RL tools ☺



[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

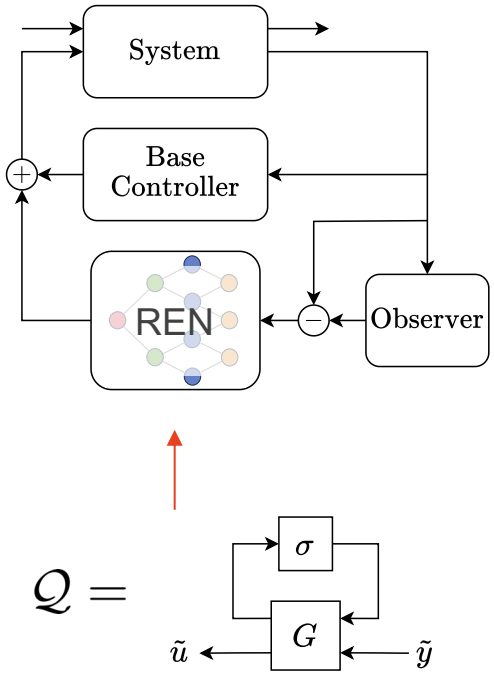
The Youla-REN

- Theory says: closed-loop stability guaranteed if we have contracting & Lipschitz Q

- Implementation: Recurrent Equlibrium Network (REN)
  - Contracting & Lipschitz
  - Universal approximator
  - Direct parametrisation

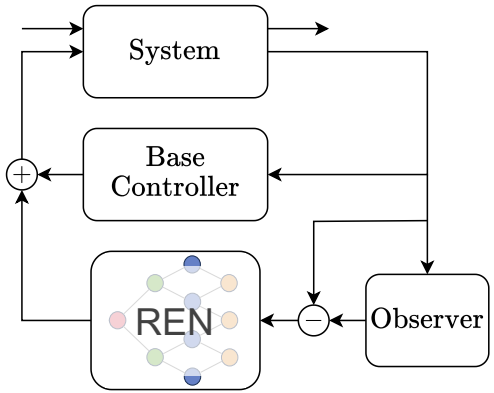
Directly learn controllers with robust stability guarantees using standard ML/deep RL tools ☺

- Practical implementation:
  - Design base controller and observer first
  - If model uncertainty, restrict gain of REN



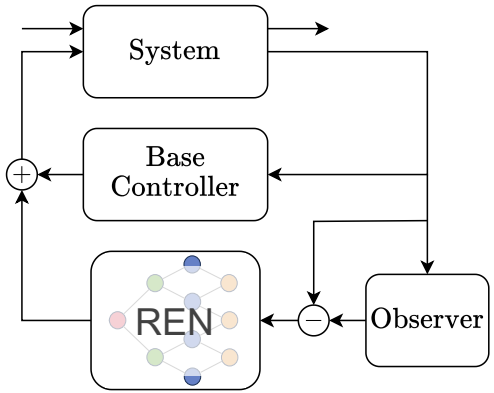
[17] Barbara et al., React to Surprises: Stable-by-Design Neural Feedback Control and the Youla-REN, *arXiv* (2025).

Learning with Stability: Example 1



Learning with Stability: Example 1

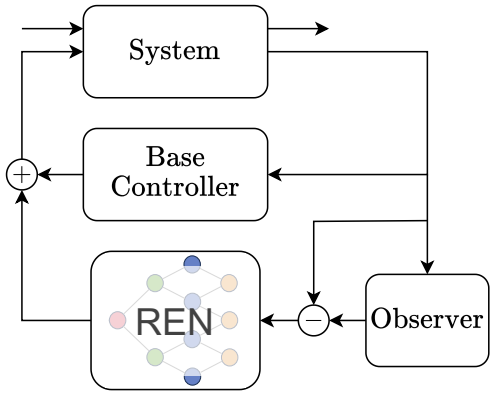
- Learning to stabilise nonlinear systems with economic costs



Learning with Stability: Example 1

- Learning to stabilise nonlinear systems with economic costs
- System dynamics

$$\dot{x}_1 = -x_1 + x_3 + w_1$$
$$\dot{x}_2 = x_1^2 - x_2 - 2x_1x_3 + x_3 + w_2$$
$$\dot{x}_3 = -x_2 + u + w_3$$
$$y = [x_2; x_3] + v$$



Learning with Stability: Example 1

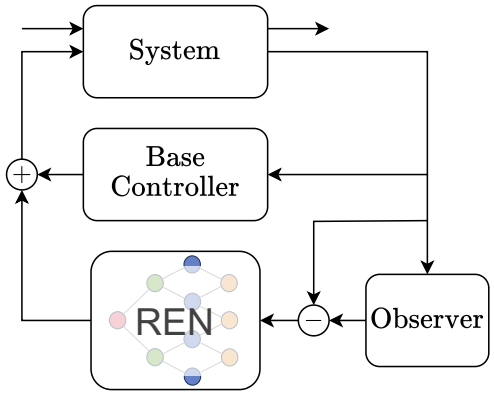
- Learning to stabilise nonlinear systems with economic costs
- System dynamics

$$\dot{x}_1 = -x_1 + x_3 + w_1$$
$$\dot{x}_2 = x_1^2 - x_2 - 2x_1x_3 + x_3 + w_2$$
$$\dot{x}_3 = -x_2 + u + w_3$$
$$y = [x_2; x_3] + v$$

- Cost function

$$\min \textstyle\sum_t \left(|u_t| + 500 \max(|u_t| - u_{\max}, 0)\right)$$

- Optimal policy is u = 0 (closed-loop unstable)



Learning with Stability: Example 1

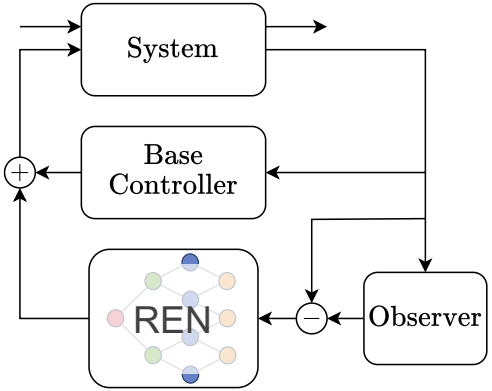
- Learning to stabilise nonlinear systems with economic costs
- System dynamics

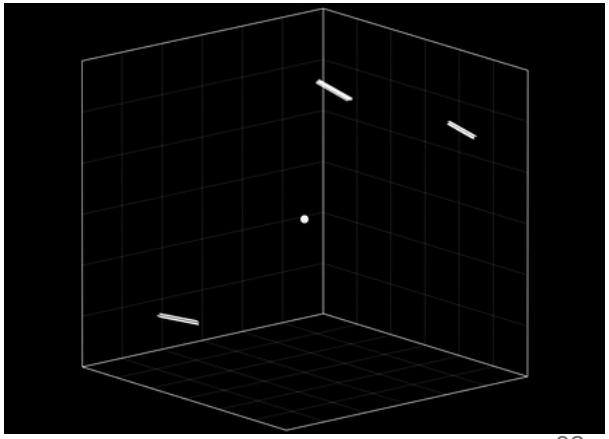
$$\dot{x}_1 = -x_1 + x_3 + w_1$$
$$\dot{x}_2 = x_1^2 - x_2 - 2x_1x_3 + x_3 + w_2$$
$$\dot{x}_3 = -x_2 + u + w_3$$
$$y = [x_2; x_3] + v$$

- Cost function

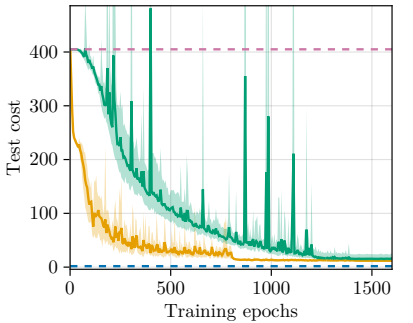
$$\min \sum_t \left(|u_t| + 500 \max(|u_t| - u_{\max}, 0)\right)$$

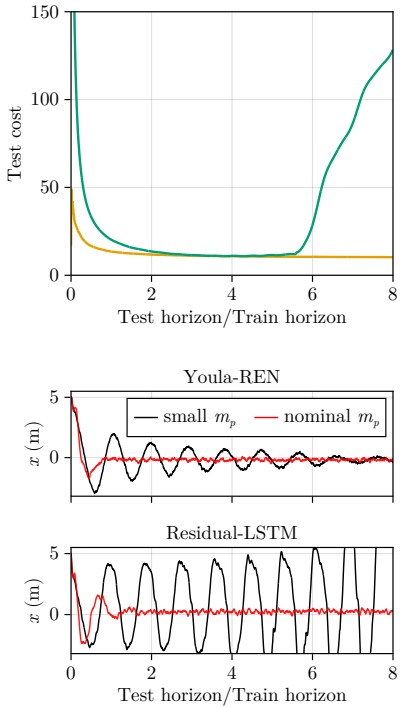
- Optimal policy is u = 0 (closed-loop unstable)





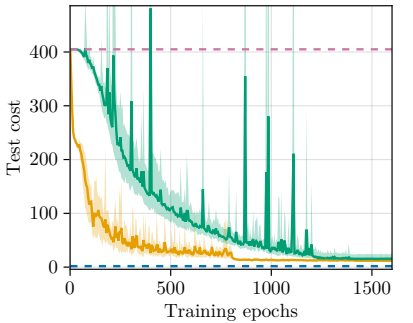
Learning with Stability: Example 2

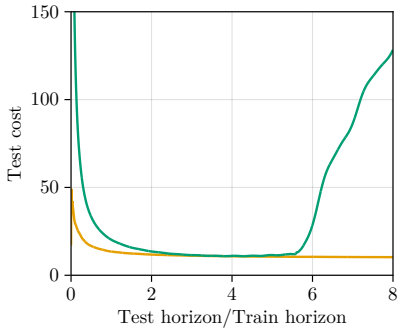


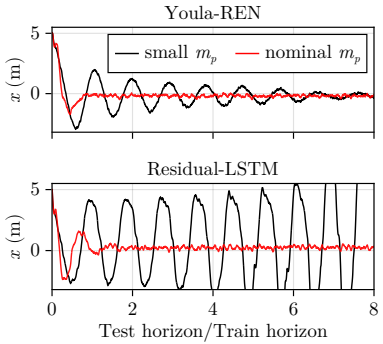


Learning with Stability: Example 2

- Linearised cart-pole, uncertain pole mass

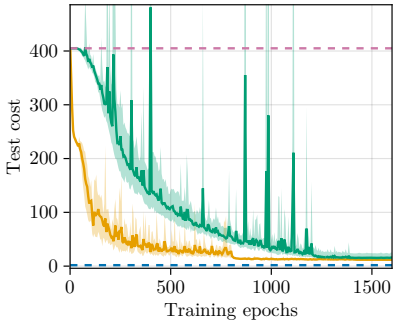


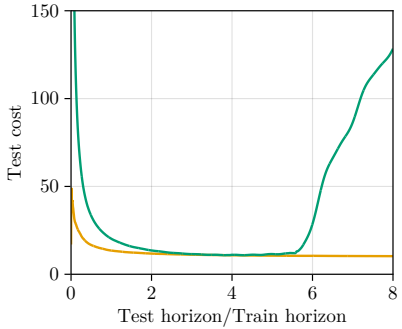


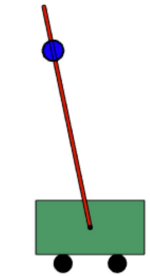


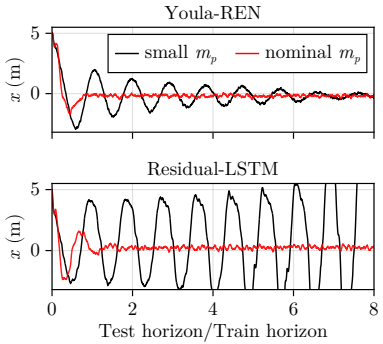
Learning with Stability: Example 2

- Linearised cart-pole, uncertain pole mass
- Youla-REN is closed-loop stable during training



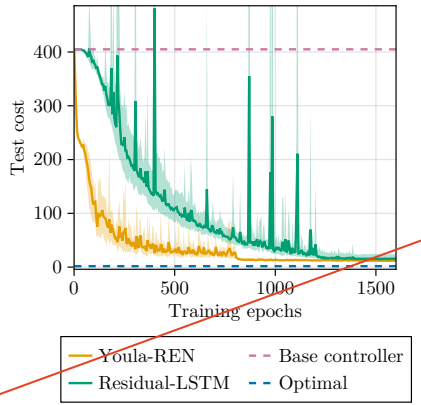


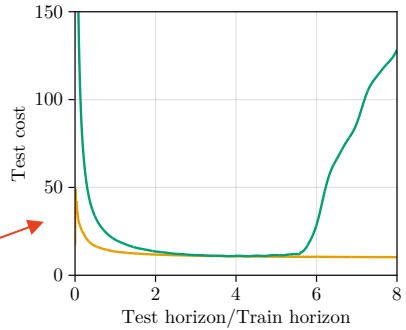


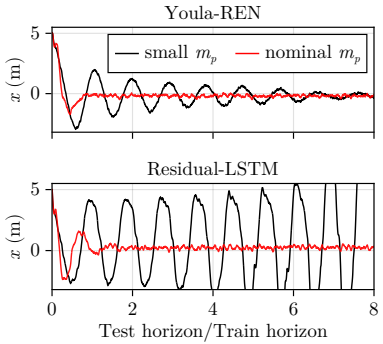


Learning with Stability: Example 2

- Linearised cart-pole, uncertain pole mass
- Youla-REN is closed-loop stable during training
- Youla-REN is closed-loop stable for long test horizons

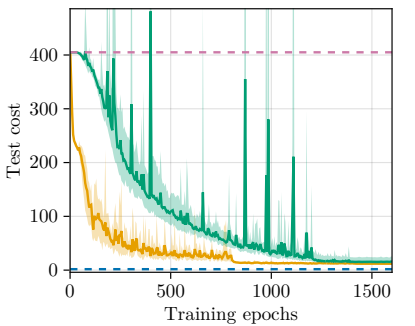


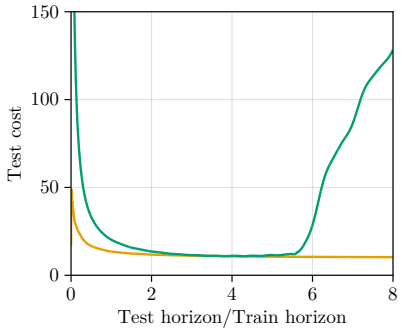


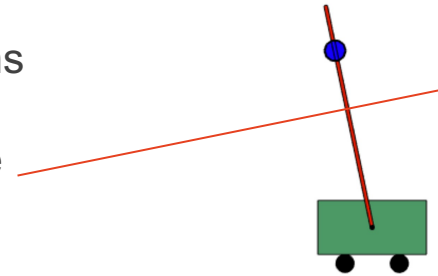


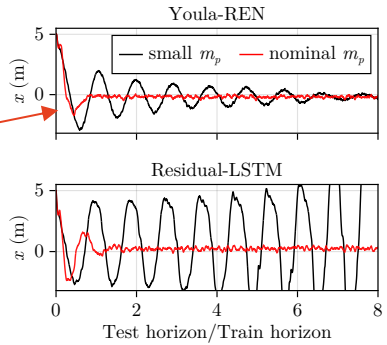
Learning with Stability: Example 2

- Linearised cart-pole, uncertain pole mass
- Youla-REN is closed-loop stable during training
- Youla-REN is closed-loop stable for long test horizons
- Youla-REN "adapts" to the uncertain parameter









Outline

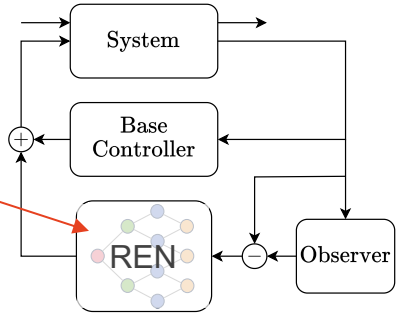
1. Motivation
2. Background: robust neural networks
3. Lipschitz-bounded policy networks
4. React to surprises with Youla-REN (Theory)
5. React to surprises with Youla-REN (Application)
6. Robust recurrent deep networks
7. Conclusions

Why Robust Recurrent Deep Network (R2DN)?

[21] Barbara et al., R2DN: Scalable Parametrization of Contracting and Lipschitz Recurrent Deep Networks, *arXiv* (2025).

Why Robust Recurrent Deep Network (R2DN)?

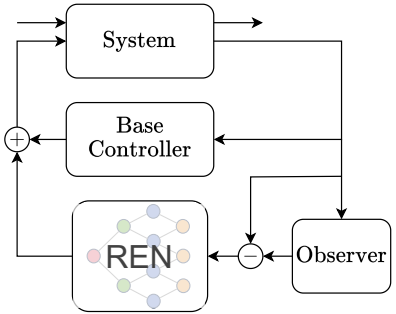
- Computational bottleneck in Youla-REN is the REN



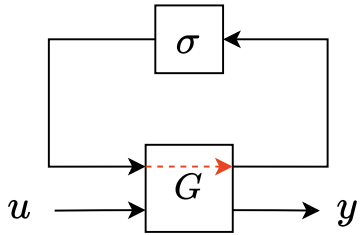
[21] Barbara et al., R2DN: Scalable Parametrization of Contracting and Lipschitz Recurrent Deep Networks, *arXiv* (2025).

Why Robust Recurrent Deep Network (R2DN)?

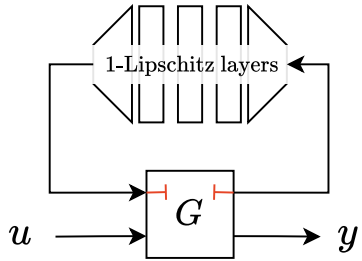
- Computational bottleneck in Youla-REN is the REN
- Introducing R2DN – like REN, but:



REN:

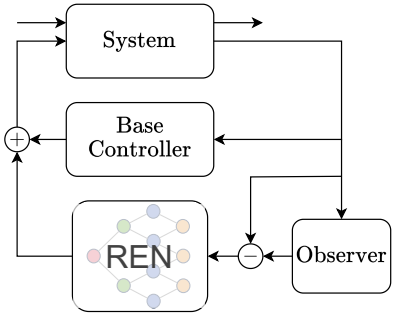


R2DN:

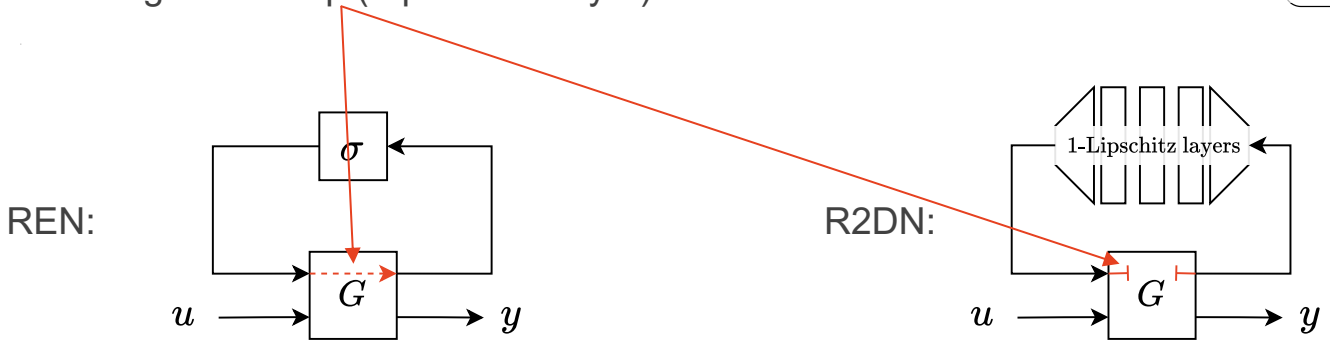


[21] Barbara et al., R2DN: Scalable Parametrization of Contracting and Lipschitz Recurrent Deep Networks, *arXiv* (2025).

Why Robust Recurrent Deep Network (R2DN)?

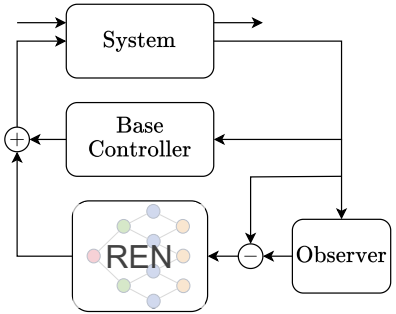


- Computational bottleneck in Youla-REN is the REN
- Introducing R2DN – like REN, but:
  - No algebraic loop (equilibrium layer)



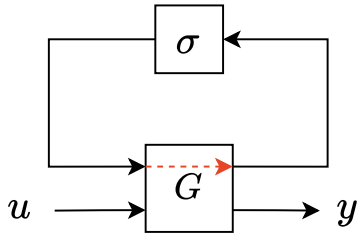
[21] Barbara et al., R2DN: Scalable Parametrization of Contracting and Lipschitz Recurrent Deep Networks, *arXiv* (2025).

Why Robust Recurrent Deep Network (R2DN)?

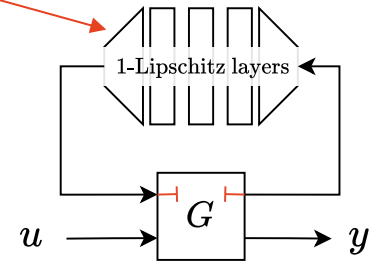


- Computational bottleneck in Youla-REN is the REN
- Introducing R2DN – like REN, but:
  - No algebraic loop (equilibrium layer)
  - Nonlinearity can be any 1-Lipschitz network

REN:

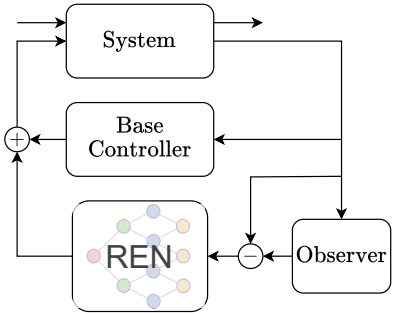


R2DN:



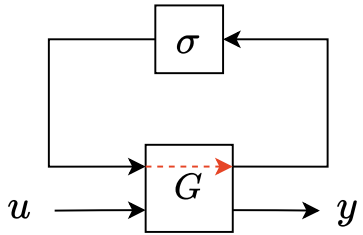
[21] Barbara et al., R2DN: Scalable Parametrization of Contracting and Lipschitz Recurrent Deep Networks, *arXiv* (2025).

Why Robust Recurrent Deep Network (R2DN)?

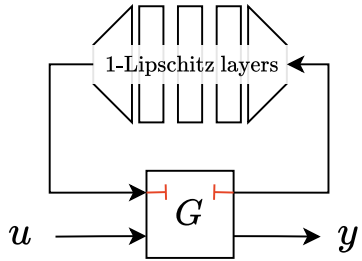


- Computational bottleneck in Youla-REN is the REN
- Introducing R2DN – like REN, but:
  - No algebraic loop (equilibrium layer)
  - Nonlinearity can be any 1-Lipschitz network

REN:



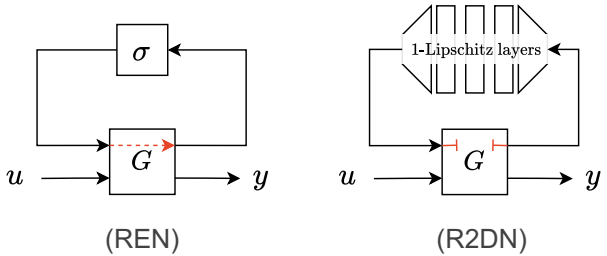
R2DN:



- Main contribution[21]: direct parametrisation of contracting & Lipschitz R2DNs

[21] Barbara et al., R2DN: Scalable Parametrization of Contracting and Lipschitz Recurrent Deep Networks, *arXiv* (2025).

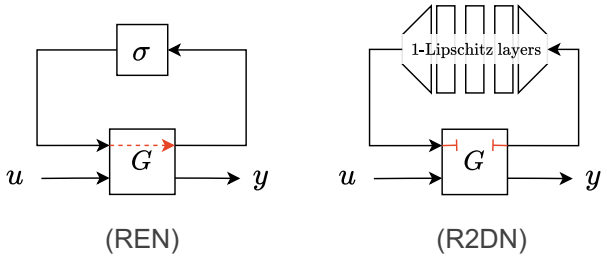
Better Scalability, Similar Performance



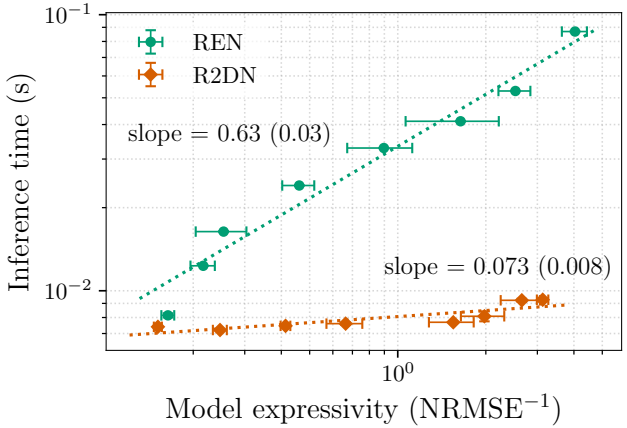
(REN) (R2DN)

Better Scalability, Similar Performance

- Main benefits:
  - No algebraic loop → efficient GPU computation
  - No algebraic loop → scalable
  - Any 1-Lipschitz network → flexible

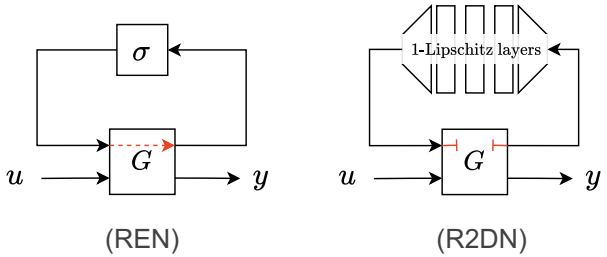


(REN) (R2DN)

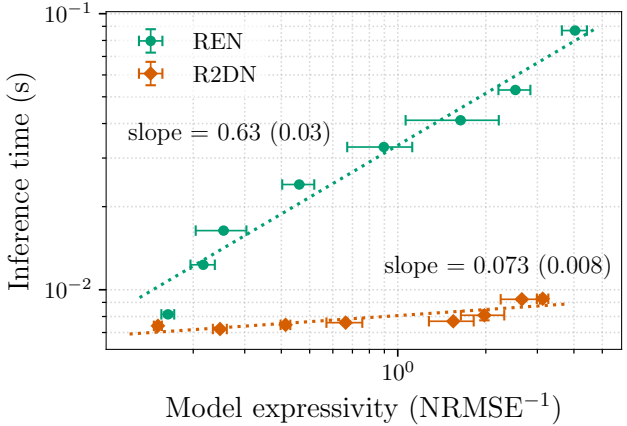


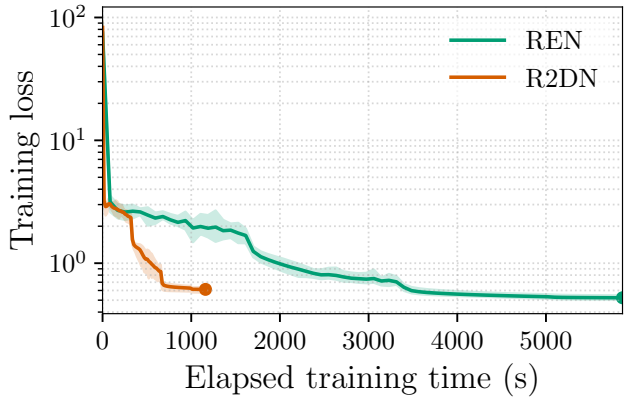
Better Scalability, Similar Performance

- Main benefits:
  - No algebraic loop → efficient GPU computation
  - No algebraic loop → scalable
  - Any 1-Lipschitz network → flexible



- Possible limitation (in theory): more conservative. In practice…





Outline

1. Motivation
2. Background: robust neural networks
3. Lipschitz-bounded policy networks
4. React to surprises with Youla-REN (Theory)
5. React to surprises with Youla-REN (Application)
6. Robust recurrent deep networks
7. Conclusions

Conclusions

- Robust NNs → learning-based control with stability & robustness guarantees

- The right choice of policy parametrisation and architecture enables:

    1. Closed-loop stability certificates
    2. Smooth, unconstrained parametrisations
    3. Non-restrictive parametrisations (under assumptions)

- “Plug-and-play” with standard deep RL and machine-learning pipelines

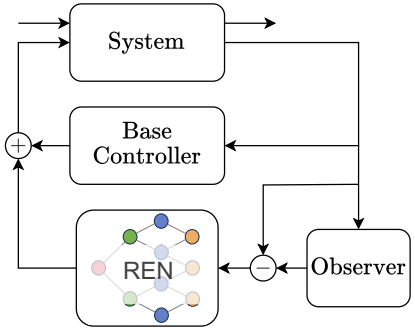
Future Work

Future Work

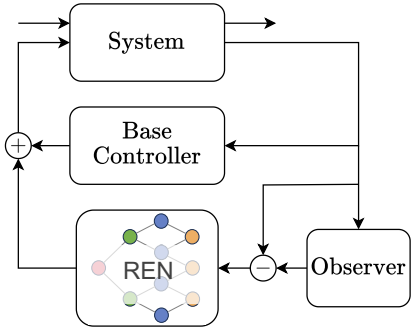
- Robust neural networks:
  - Develop more scalable parametrisations
  - Bigger library of behavioural properties & network architectures

Future Work

- Robust neural networks:
  - Develop more scalable parametrisations
  - Bigger library of behavioural properties & network architectures
- Youla parametrisation:
  - Local stability
  - Non-smooth systems
  - Prescribed robustness
  - Designing good observers?



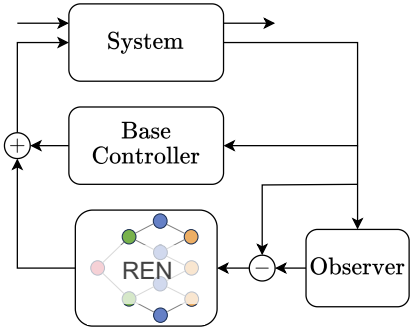
Future Work

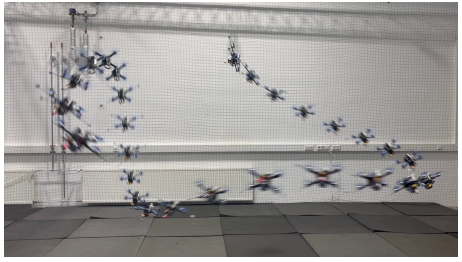


- Robust neural networks:
  - Develop more scalable parametrisations
  - Bigger library of behavioural properties & network architectures
- Youla parametrisation:
  - Local stability
  - Non-smooth systems
  - Prescribed robustness
  - Designing good observers?
- Interplay between architectures and their parametrisations

Future Work

- Robust neural networks:
  - Develop more scalable parametrisations
  - Bigger library of behavioural properties & network architectures
- Youla parametrisation:
  - Local stability
  - Non-smooth systems
  - Prescribed robustness
  - Designing good observers?
- Interplay between architectures and their parametrisations
- Applications to robotics[22]





[22] Pries & Ryll, Learning Robust Agile Flight Control with Stability Guarantees, *arXiv* (2025).