dortmund university

Efficient approximation of model predictive control via neural networks with guarantees

B. Karg, M. Heinlein, J. Adamek, L. Lüken, D. Brandner and S. Lucia
Chair of Process Automation Systems
Department of Biochemical and Chemical Engineering

Outline

- Approximate MPC with neural networks
- Efficient sampling via Sobolov Training
- Deterministic and probabilistc guarantees

Outline

- Approximate MPC with neural networks
- Efficient sampling via Sobolov Training
- Deterministic and probabilistc guarantees

Towards complex autonomous systems

Huge Challenges

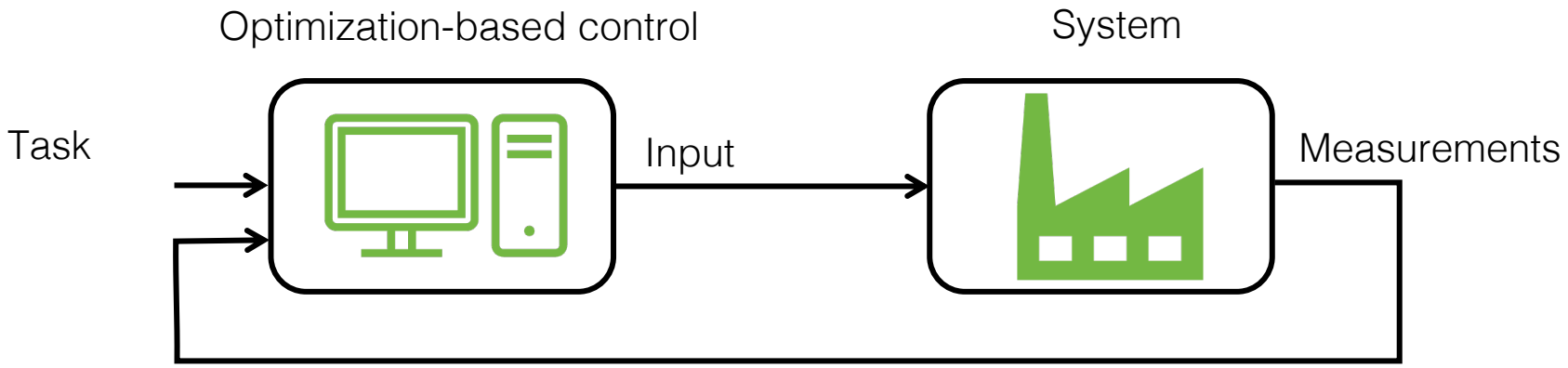
- Interactions between many subsystems
- Scalability
- Optimality
- Safety & security
- Validation & verification

Optimization-based operation and control

Optimization of processes is a major engineering goal

- Improved performance (energy, production volume, emissions)
- Subject to quality, safety and legal requirements

Optimization-based control to achieve this task!



One of the major trends in control in the last 30 years

Mathematical formulation of MPC

control goal

$$\underset{u_k}{\text{minimize}} \quad \sum_{k=0}^{N-1} \ell(x_k, u_k) + V_f(x_N)$$

$$\text{subject to} \quad x_{k+1} = f(x_k, u_k) \quad \text{system model}$$

$$g(x_k, u_k) \leq 0, \quad \text{constraints}$$

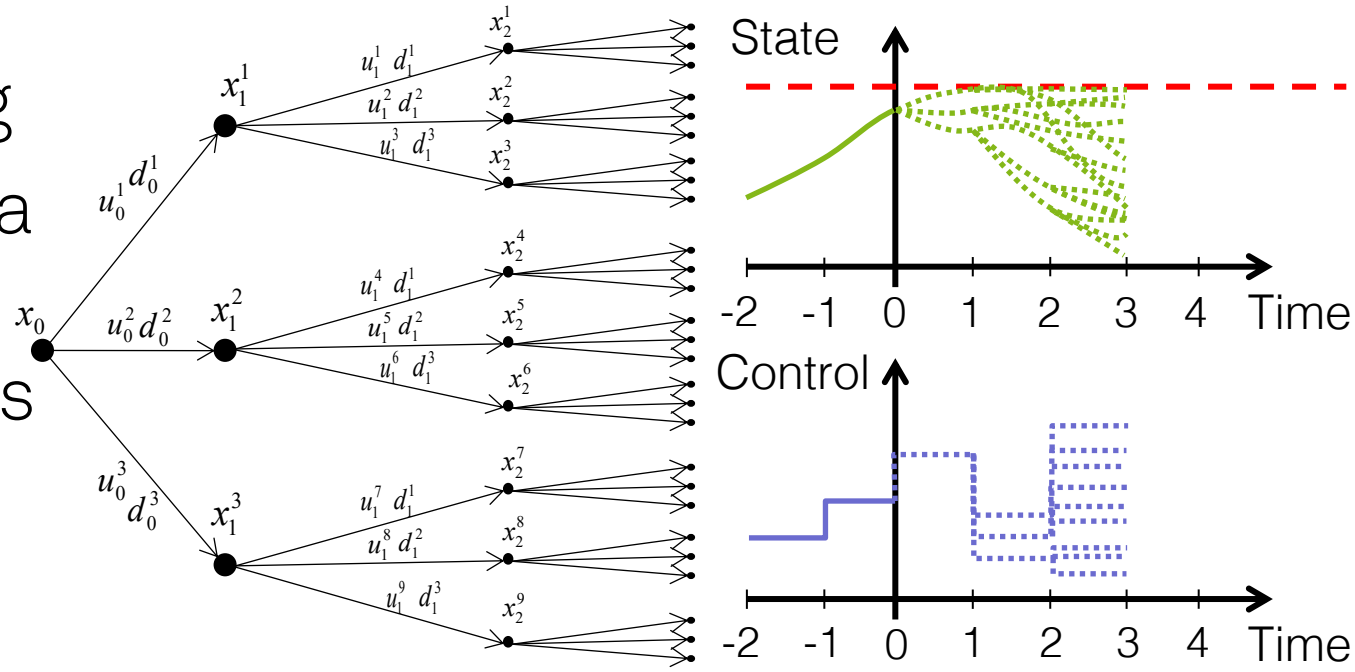
$$x_0 = x_{\text{init}}, \quad \text{system state}$$

$$k \in [0, N-1]$$

Considering uncertainties in MPC

In practice, all models are wrong

- Example: exact parameters of a model typically not available
- Can we make optimal decisions that respect constraints despite model errors?
- Yes, at the cost of increasing computational complexity



Proposed in the linear case in [Scokaert et al., 1998], [de la Peña et al., 2005]
Developed in the nonlinear case mostly in [Lucia et al., JPC, 2013, 2014; SCL 2020]
Many extensions by several groups [D. Krishnamoorthy et al., IEEE CSL 2018, …]

Solving complex problems reliably in real time

Solving optimization-based control problems in real time is still challenging:

- For very fast systems
- For complex systems
- On simple / standard hardware

Even more challenging if uncertainty is present

- **Goal:** Development of an approach that simultaneously
  - Obtains approximate optimal robust solutions
  - Can be computed reliably even on standard industrial devices

Main idea

Most optimization-based decision-making problems are **parametric**

$$
\begin{aligned}
\underset{u_k}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \ell(x_k, u_k) + V_f(x_N) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k) \\
& g(x_k, u_k) \leq 0, \\
& x_0 = x_{\text{init}}, \qquad \text{system state} \\
& k \in [0, N-1]
\end{aligned}
$$

An implicit mapping $x_{\text{init}} \rightarrow u^*$ is defined by the optimization problem

- Is it possible to learn this mapping using machine learning techniques?

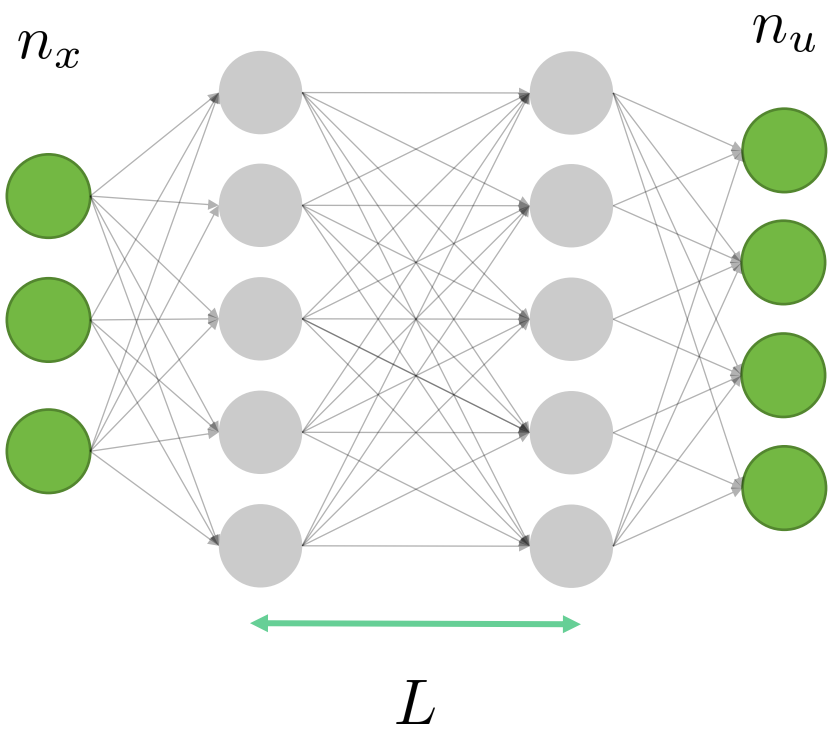
Using neural networks to approximate MPC

It is an old idea: already done in 1995 for nonlinear MPC

- What is new?

Common practice until recent successes in deep learning was:

- Because of **universal approximation theorem**: use only 1 layer



[T. Parisini and R. Zoppoli, 1995, Akesson and Toivonen, 2006]

Deep neural networks (DNN)

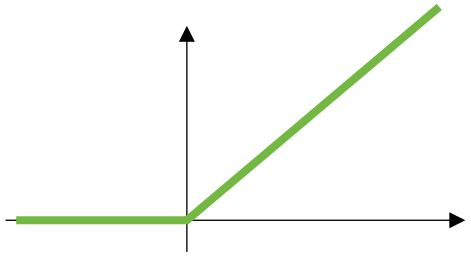
Neural network with L **hidden layers** and M neurons per layer

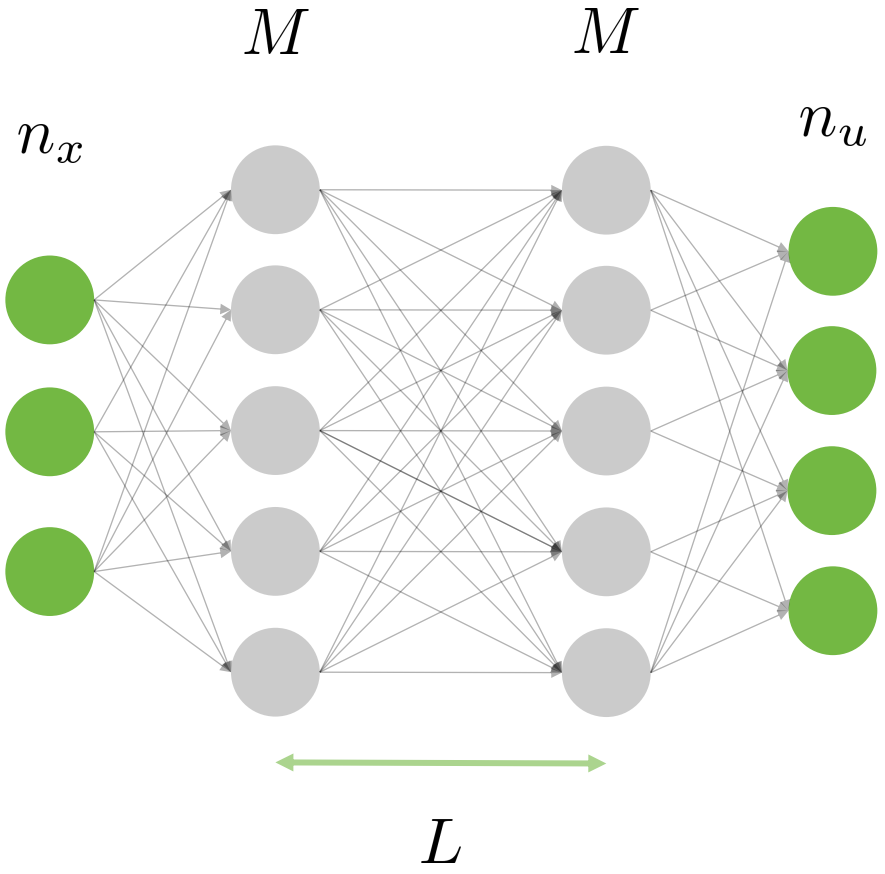
$$\mathcal{N}(x;\theta,M,L) = f_{L+1} \circ g_L \circ f_L \circ \cdots \circ g_1 \circ f_1(x)$$

- Affine transformation $\theta_l = \{W_l, b_l\}$

  $f_l(x_{l-1}) = W_l x_{l-1} + b_l$

- Activation function
  - tanh: $g_l(f_l) = \tanh(f_l) = \dfrac{e^{f_l} - e^{-f_l}}{e^{f_l} + e^{-f_l}}$
  - ReLU: $g_l(f_l) = \max(0, f_l)$

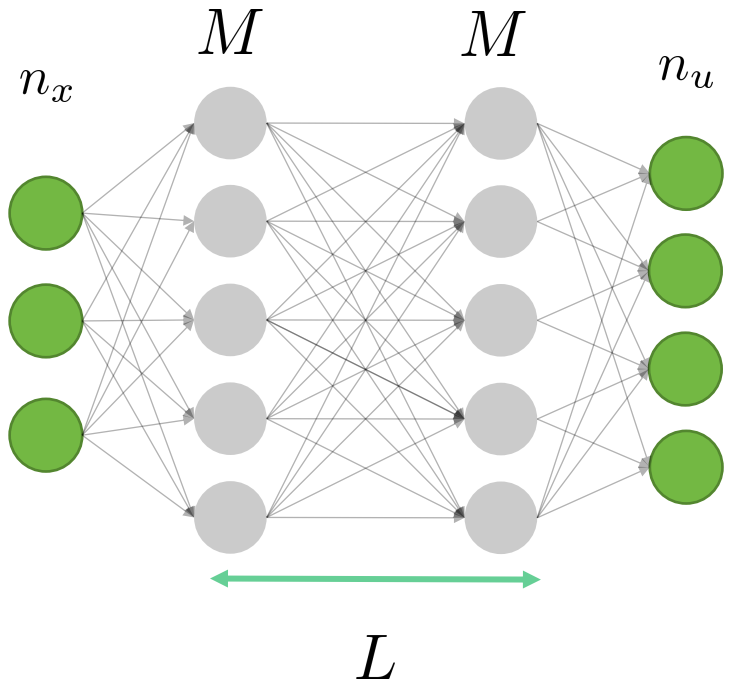




Why deep neural networks to approximate MPC?

- One of the main questions in machine learning
- An intuitition why more hidden layers can help:
  - The number of linear regions represented by a network grows exponentially with the number of layers [Montufar 2014], [Chen et al, 2018; Karg and Lucia, 2018]

$$n_{\mathrm{r}} = \left(\prod_{l=1}^{L-1} \left\lfloor \frac{M}{n_x} \right\rfloor^{n_x} \right) \sum_{j=0}^{n_x} \binom{L}{j}$$



- It is also possible to compute the number of neurons and layers necessary to exactly represent the solution of an MPC problem

[Karg and Lucia, IEEE Transactions on Cybernetics, 2020]

Theorem 1. *There always exist parameters $\theta_{\gamma,i}$ and $\theta_{\eta,i}$ for $2n_u$ deep ReLU neural networks with depth $r_{\gamma,i}$ and $r_{\eta,i}$ for $i = 1, \ldots, n_u$ and width $M = n_x + 1$, such that the vector of neural networks defined by*

$$\begin{bmatrix} \mathcal{N}(x; \theta_{\gamma,1}, M, r_{\gamma,1}) - \mathcal{N}(x; \theta_{\eta,1}, M, r_{\eta,1}) \\ \vdots \\ \mathcal{N}(x; \theta_{\gamma,n_u}, M, r_{\gamma,n_u}) - \mathcal{N}(x; \theta_{\eta,n_u}, M, r_{\eta,n_u}) \end{bmatrix} \quad (15)$$

can exactly represent an explicit MPC law $\mathcal{K}(x) : [0,1]^{n_x} \to \mathbb{R}^{+n_u}$.

Exact representation

It is possible to exactly represent a linear MPC law with a ReLU neural network

- Divide the PWA function in a difference of two convex PWA functions with r_γ, r_η regions

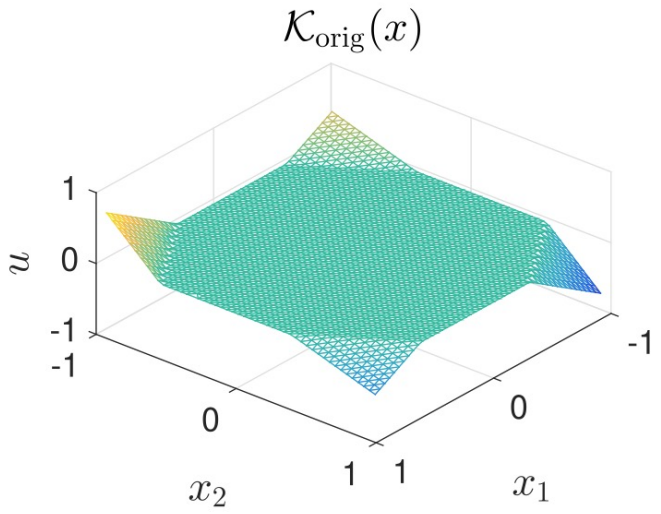
$$f(x) = \gamma(x) - \eta(x)$$

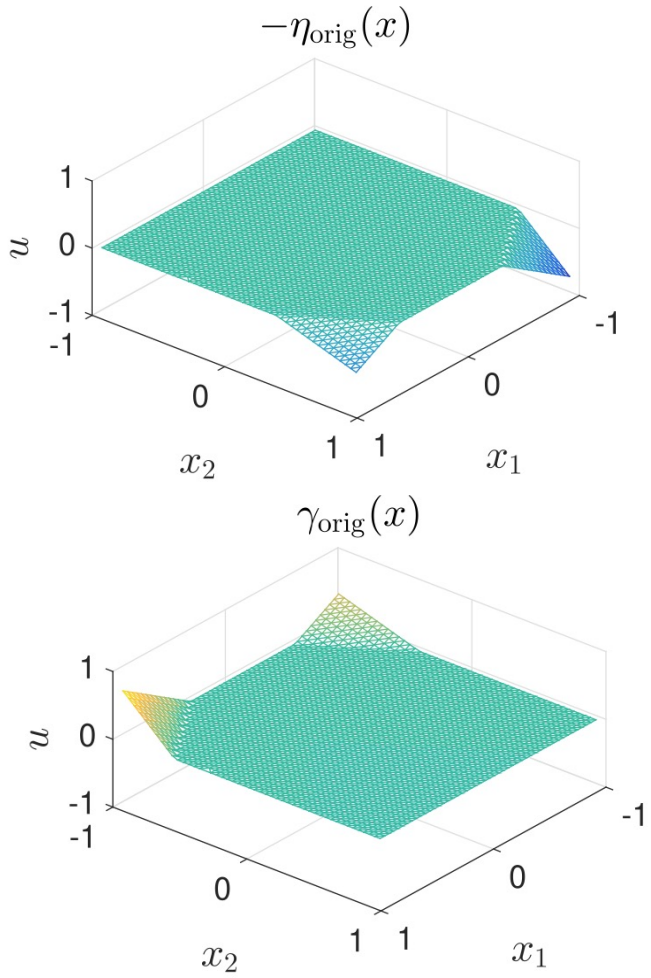
- Each convex PWA function can be represented by the difference of two ReLU network of width $M = n_x + 1$ and depth r_γ, r_η

Main problem: in general, still exponential growth of r_γ, r_η

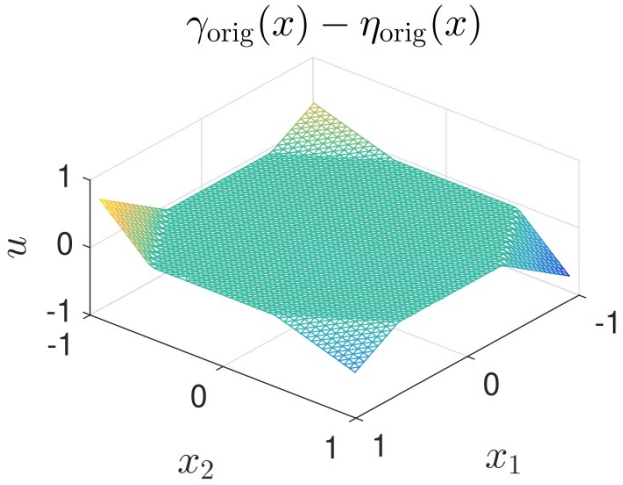
[Karg and Lucia, IEEE Transactions on Cybernetics, 2020]

Exact representation: Example





Exact representation via NNs



3 layers each,
3 neurons per layer

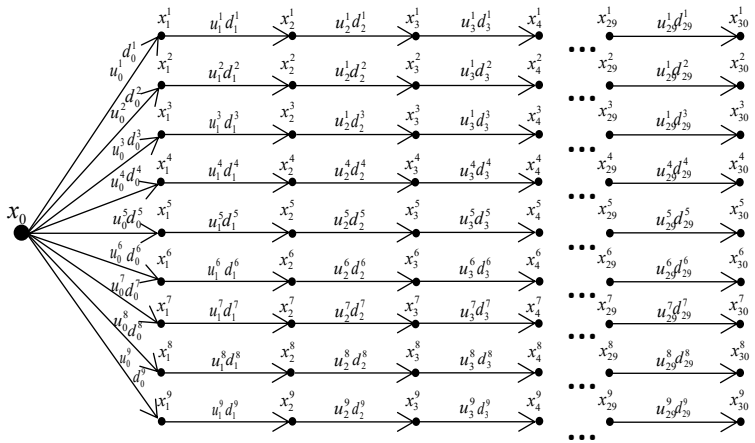
[Karg and Lucia, IEEE Transactions on Cybernetics, 2020]

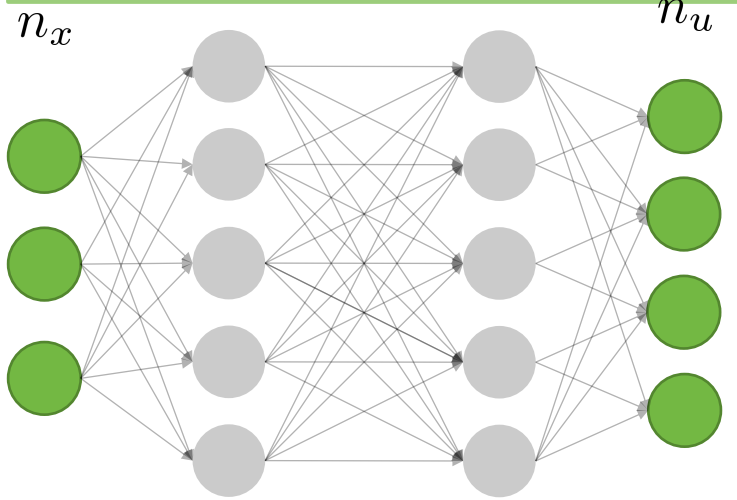
Approximating a complex controller

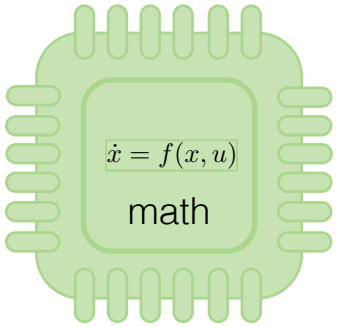
1: Generate training samples by solving many MPC problems

(x_{init}, u_0^*)

2: Offline training of the deep neural network

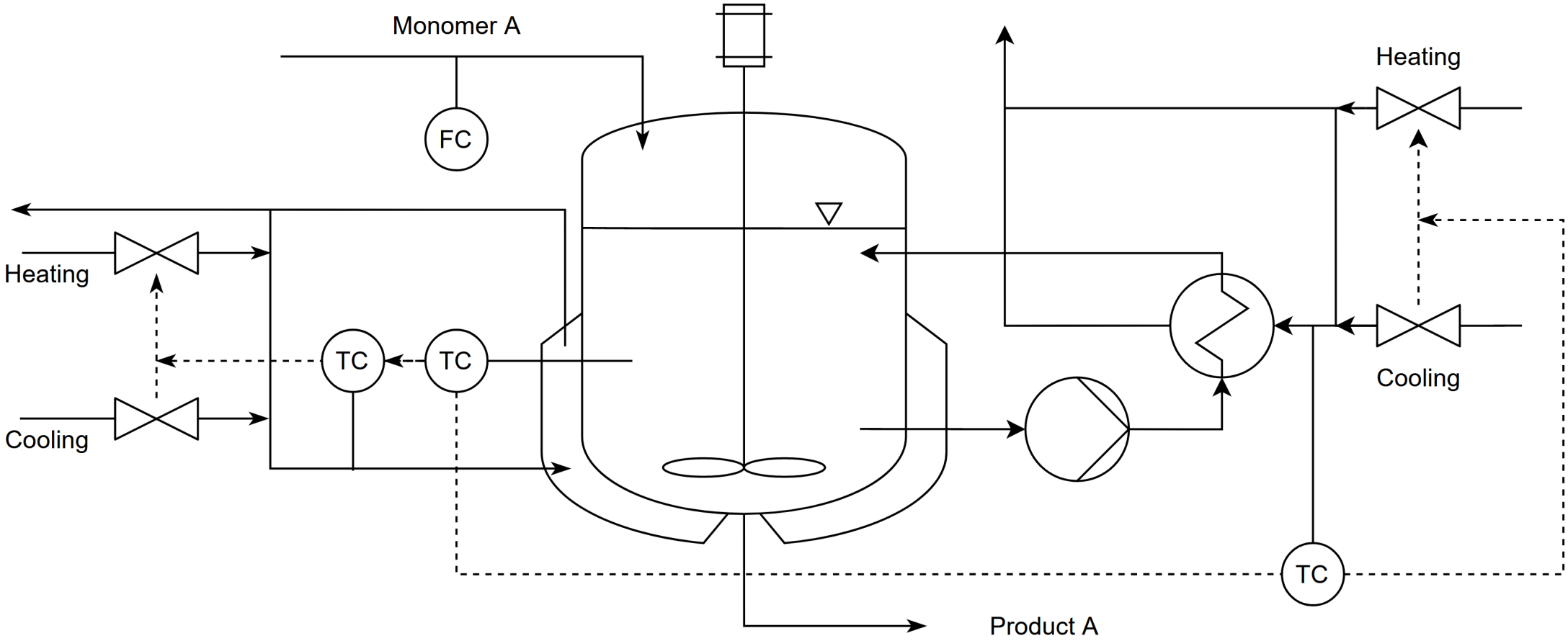






3: High performance control even on simple hardware

Case-study: Industrial polymerization reactor



[Lucia et al., *Journal of Process Control, 2014*]

An industrial polymerization reactor

8 differential states
3 control inputs
2 uncertain parameters

$$\dot{m}_W = \dot{m}_{W,F}$$

$$\dot{m}_A = \dot{m}_{A,F} - k_{R1} m_{A,R} - \frac{p_1 k_{R2} m_{AWT} m_A}{m_{ges}}$$

$$\dot{m}_P = k_{R1} m_{A,R} + \frac{p_1 k_{R2} m_{AWT} m_A}{m_{ges}}$$

$$\dot{T}_R = \frac{1}{c_{p,R} m_{ges}} \left[\dot{m}_F c_{p,F}(T_F - T_R) + \Delta H_R k_{R1} m_{A,R} - k_K A(T_R - T_S) - \dot{m}_{AWT} c_{p,R}(T_R - T_{EK})\right]$$

$$\dot{T}_S = 1/(c_{p,S} m_S)[k_K A(T_R - T_S) - k_K A(T_S - T_M)]$$

$$\dot{T}_M = \frac{1}{c_{p,W} m_{M,KW}} \left[\dot{m}_{M,KW} c_{pW}(T_M^{IN} - T_M) + k_K A(T_S - T_M)\right]$$

$$\dot{T}_{EK} = \frac{1}{c_{p,R} m_{AWT}} \left[\dot{m}_{AWT} c_{p,W}(T_R - T_{EK}) - \alpha(T_{EK} - T_{AWT}) + \frac{p_1 k_{R2} m_A m_{AWT} \Delta \mathrm{H}_R}{m_{ges}}\right]$$

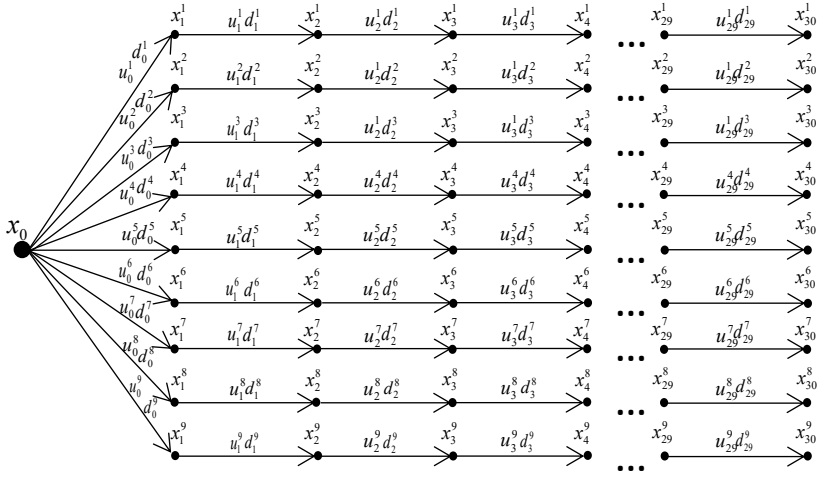
$$\dot{T}_{AWT} = \frac{1}{c_{p,W} m_{AWT,KW}} \left[\dot{m}_{AWT,KW} c_{p,W}(T_{AWT}^{IN} - T_{AWT}) - \alpha(T_{AWT} - T_{EK})\right]$$

$$k_{R1} = k_0 e^{-\frac{E_a}{R T_R}} (k_{U1}(1 - U) + k_{U2} U)$$

$$k_{R2} = k_0 e^{-\frac{E_a}{R T_{EK}}} (k_{U1}(1 - U) + k_{U2} U)$$

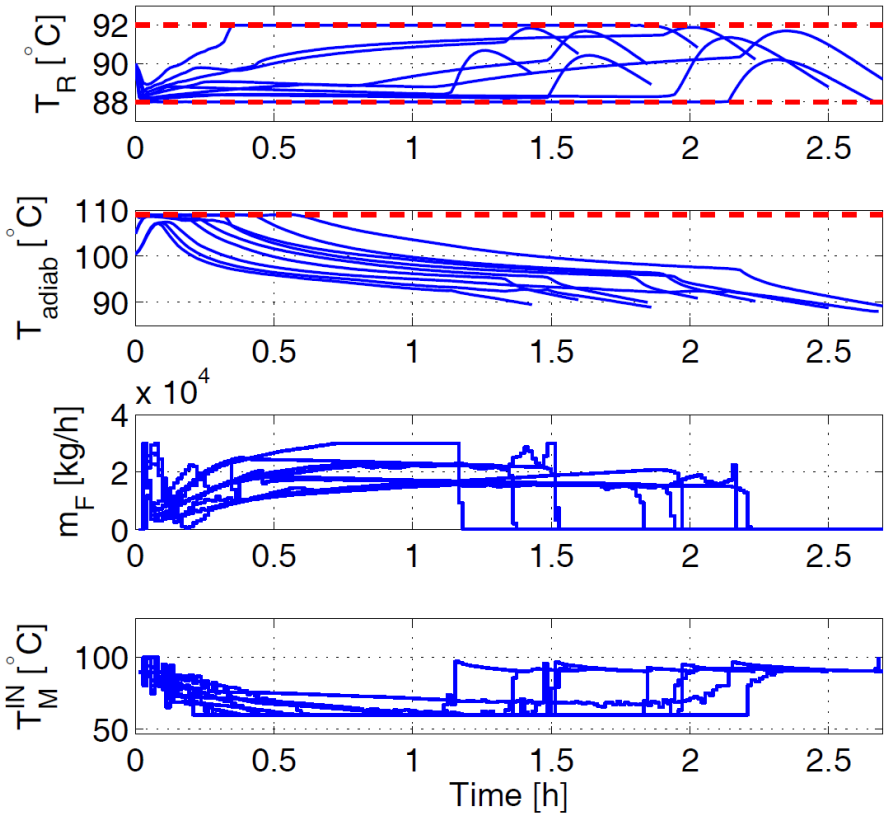
Simulation results for multi-stage NMPC

- A robust MPC is possible by considering different values of critical uncertainty parameters in a simple scenario tree



- At the cost of increased computational cost

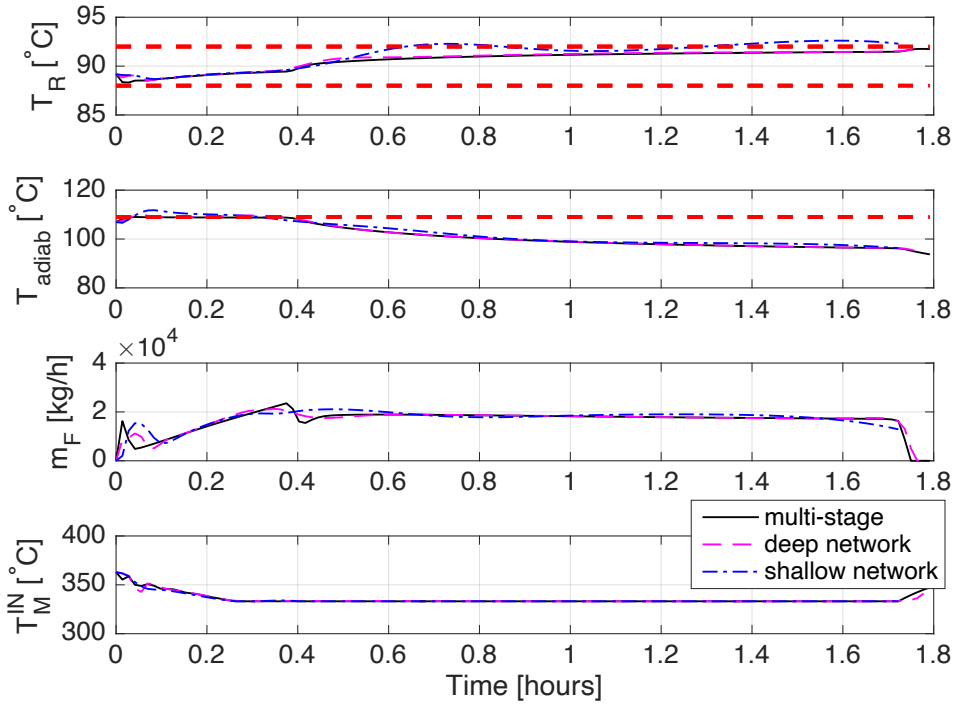
Multi-stage NMPC



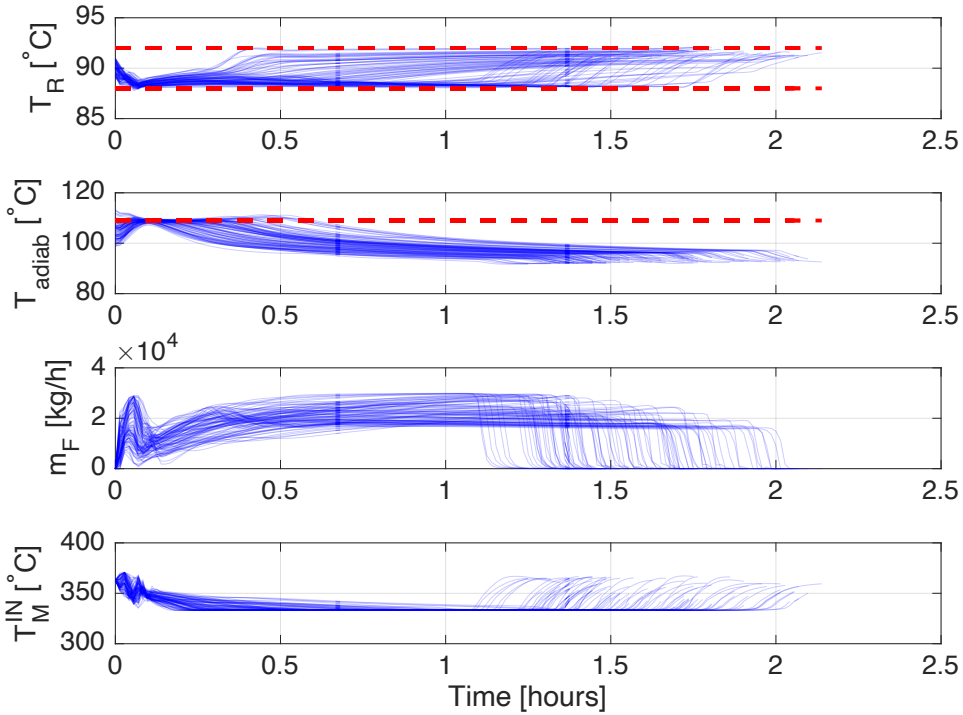
Can we learn such controller?

Performance of deep-learning based ms-NMPC

Exact vs. deep vs. shallow multi-stage NMPC



Deep-learning based multi-stage NMPC



Average performance over random 100 batches

Algorithm	Batch time [h]	Cons. Viol. [°C/h]
Exact	1.6459	0.0058
Shallow	1.7328	0.3087
Deep	1.6297	0.0549

[Lucia and Karg, NMPC 2018]

A batch polymerization reactor

Enable low-cost emb. implementation

- 32 bit ARM Cortex M0+
- 48 MHz with 32 kB RAM

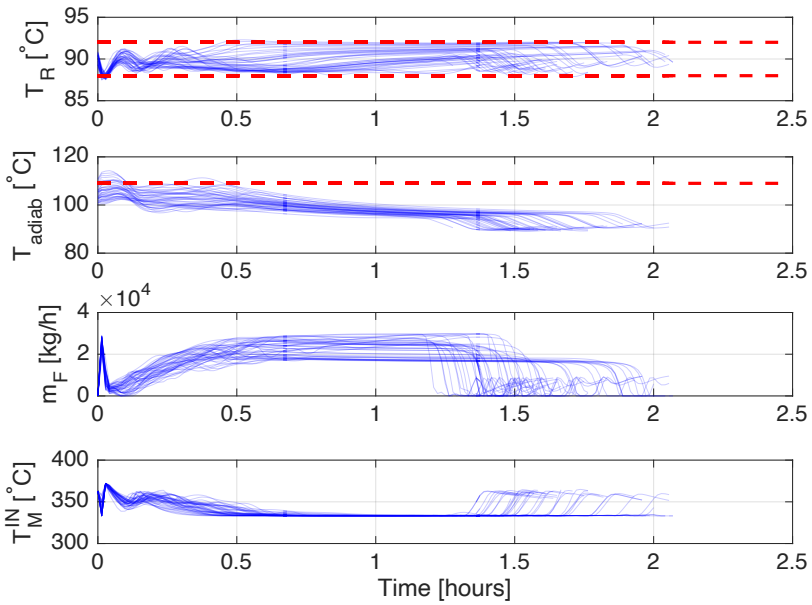
Approx. robust NMPC:

- Memory footprint: 27 kB
- Evaluation time on a uC: 37 ms
- Trivial code-generation (PLC, uC)

Enable large(r)-scale problems

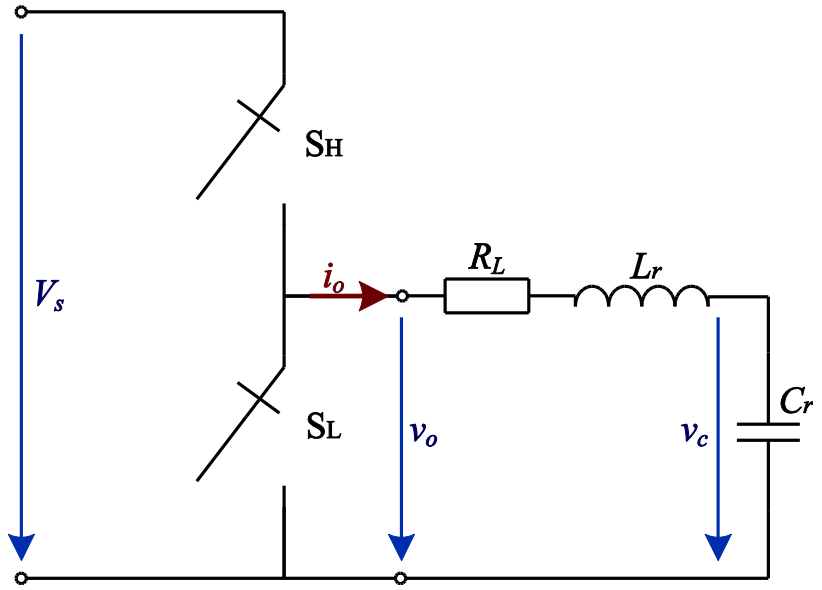
Problem with 5 uncertainties

- 243 scenarios
- ~115,000 variables and constraints



Robust nonlinear optimal control in microseconds

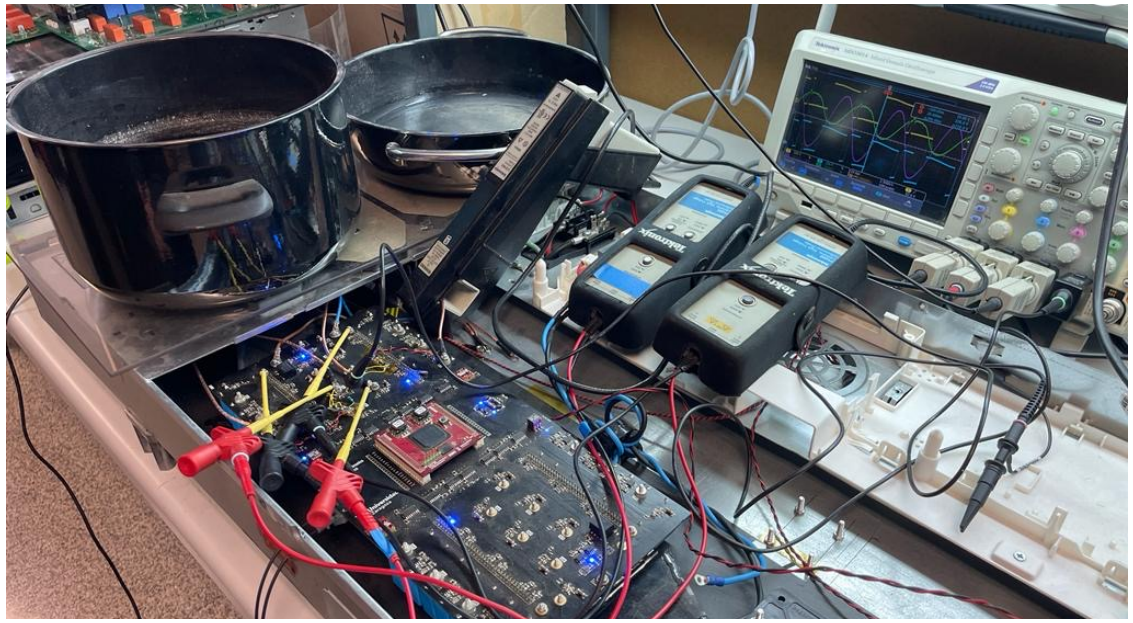
Induction heating is currently used in many industrial and domestic applications



Control switching frequency and duty cycle. Satisfy constraints under uncertainty

Real implementation

Approximate robust NMPC in 1 μs (on an FPGA)



[S. Lucia et al., IEEE Transactions on Industrial Informatics, 2020]

[P. Guillen et al., IEEE Access, 2022]

Universidad Zaragoza 1542

Outline

- Approximate MPC with neural networks
- Efficient sampling via Sobolov Training
- Deterministic and probabilistc guarantees

How difficult is it to sample properly?

Data-Efficient Approximate NMPC

- Augmented loss using knowledge about constraints[1]
- NLP sensitivities for data augmentation[2]
- Fitting gradients of linear MPC control law[3]

➢Use of inherent NLP information

All details in [Lüken, Brandner and Lucia, IFAC WC 2023]

- This work
  - Use of parametric NLP sensitivities
  - Fitting gradients: Sobolev training

$$\min_{\boldsymbol{\Theta}} \; \mathcal{L}_{\text{nom.}}(\boldsymbol{x}_0, \boldsymbol{u}_0; \boldsymbol{\Theta}) + \left\| \frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{x}_0}(\boldsymbol{x}_0) - \frac{\partial \hat{\boldsymbol{u}}_0}{\partial \boldsymbol{x}_0}(\boldsymbol{x}_0; \boldsymbol{\Theta}) \right\|_{\text{F}}^2$$

- Extension of Ideas of[2,3]

➢How to obtain sensitivities?

➢How to train NN with sensitivities?

[1] S. Adhau, V. Naik, S. Skogestad, 2021, CDC
[2] D. Krishnamoorthy, 2022, TAC
[3] R. Winqvist, A. Venkitaraman, B. Wahlberg, 2021, IFAC Papers Online

Sobolev Training[1,2]

All details in [Lüken, Brandner and Lucia, IFAC WC 2023]

- „Sobolev Spaces“:
  - Metric spaces containing derivatives of function[1]
- Data

$$\mathbb{D} = \{(\boldsymbol{x}_0^i, \boldsymbol{u}_0^i, \left(\frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{x}_0}\right)^i) | i = 1, \ldots, N_s\}$$

- Predictions

$$\hat{\boldsymbol{u}}_0^i = NN(\boldsymbol{x}_0^i; \boldsymbol{\Theta}) \in \mathbb{R}^{n_u}$$

$$\left(\frac{\partial \hat{\boldsymbol{u}}_0}{\partial \boldsymbol{x}_0}\right)^i = \frac{\partial NN}{\partial \boldsymbol{x}_0}(\boldsymbol{x}_0^i; \boldsymbol{\Theta}) \in \mathbb{R}^{n_u \times n_x}$$

- Sobolev Loss

$$\mathcal{L}_{\text{sob}}(\boldsymbol{x}_0, \boldsymbol{u}_0, \frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{x}_0}; \boldsymbol{\Theta}) = \frac{1}{N_s} \sum_{i=1}^{N_s} \|\boldsymbol{u}_0^i - \hat{\boldsymbol{u}}_0^i\|_2^2 + \gamma \cdot \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| \left(\frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{x}_0}\right)^i - \left(\frac{\partial \hat{\boldsymbol{u}}_0}{\partial \boldsymbol{x}_0}\right)^i \right\|_{\text{F}}^2$$

- Training (via SGD)

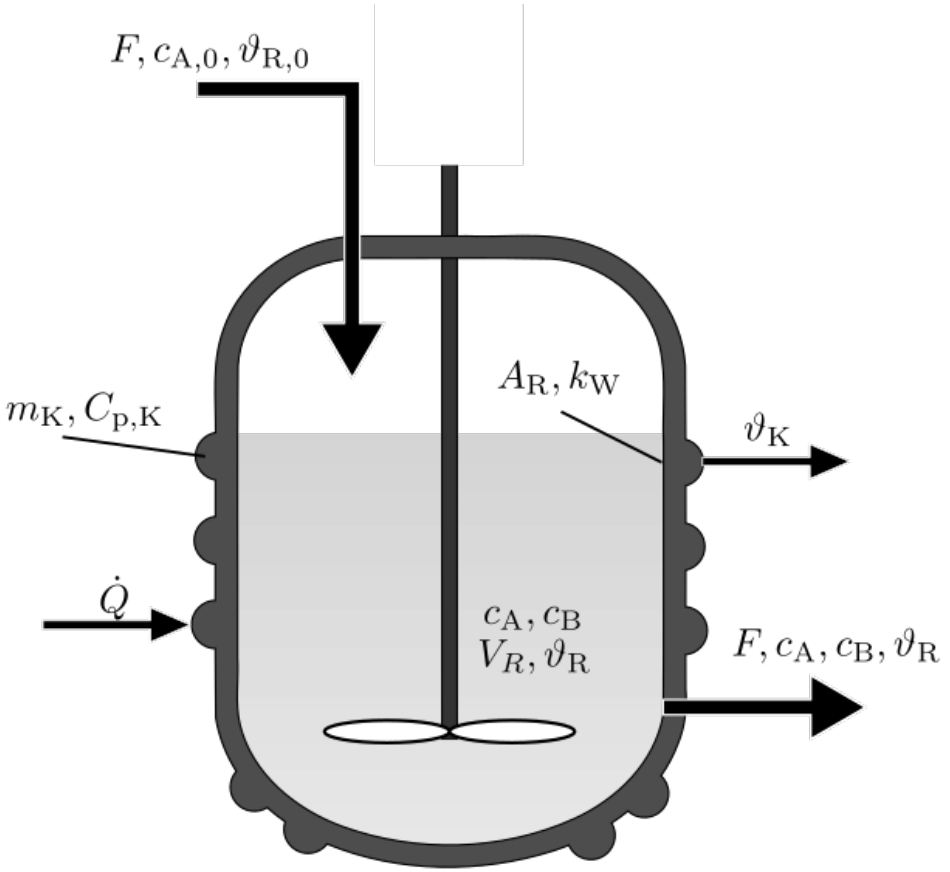
$$\min_{\boldsymbol{\Theta}} \mathcal{L}_{\text{sob}}(\boldsymbol{x}_0, \boldsymbol{u}_0, \frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{x}_0}; \boldsymbol{\Theta})$$

[1] W. Czarnecki et al., 2017, NeurIPS
[2] J. Cocola and P. Hand, 2020, LOD

Continuous stirred tank reactor (CSTR)

- Nonlinear dynamics
- 4 states $(c_\mathrm{A}, c_\mathrm{B}, \vartheta_\mathrm{R}, \vartheta_\mathrm{K})$
- 2 control inputs $\left(F, \dot{Q}\right)$

$$
\begin{aligned}
\frac{\partial c_\mathrm{A}}{\partial t} &= F\left(c_{\mathrm{A},0} - c_\mathrm{A}\right) - k_1 c_\mathrm{A} - k_3 c_\mathrm{A}^2 \\
\frac{\partial c_\mathrm{B}}{\partial t} &= -F c_\mathrm{B} + k_1 c_\mathrm{A} - k_2 c_\mathrm{B} \\
\frac{\partial \vartheta_\mathrm{R}}{\partial t} &= F\left(\vartheta_{\mathrm{R},0} - \vartheta_\mathrm{R}\right) + \frac{k_\mathrm{W} A_\mathrm{R}}{\varrho C_{p,\mathrm{R}} V_\mathrm{R}}\left(\vartheta_\mathrm{K} - \vartheta_\mathrm{R}\right) \\
&\quad - \frac{k_1 c_\mathrm{A} \Delta H_{R,1} + k_2 c_\mathrm{B} \Delta H_{R,2} + k_3 c_\mathrm{A}^2 \Delta H_{R,3}}{\varrho C_{p,\mathrm{R}}} \\
\frac{\partial \vartheta_\mathrm{K}}{\partial t} &= \frac{\dot{Q} + k_\mathrm{W} A_\mathrm{R}\left(\vartheta_\mathrm{R} - \vartheta_\mathrm{K}\right)}{m_\mathrm{K} C_{p,\mathrm{K}}} \\
k_i &= k_{0,i} \exp\left(\frac{-E_{A,i}}{R\left(\vartheta_\mathrm{R} + 273.15\right)}\right)
\end{aligned}
$$

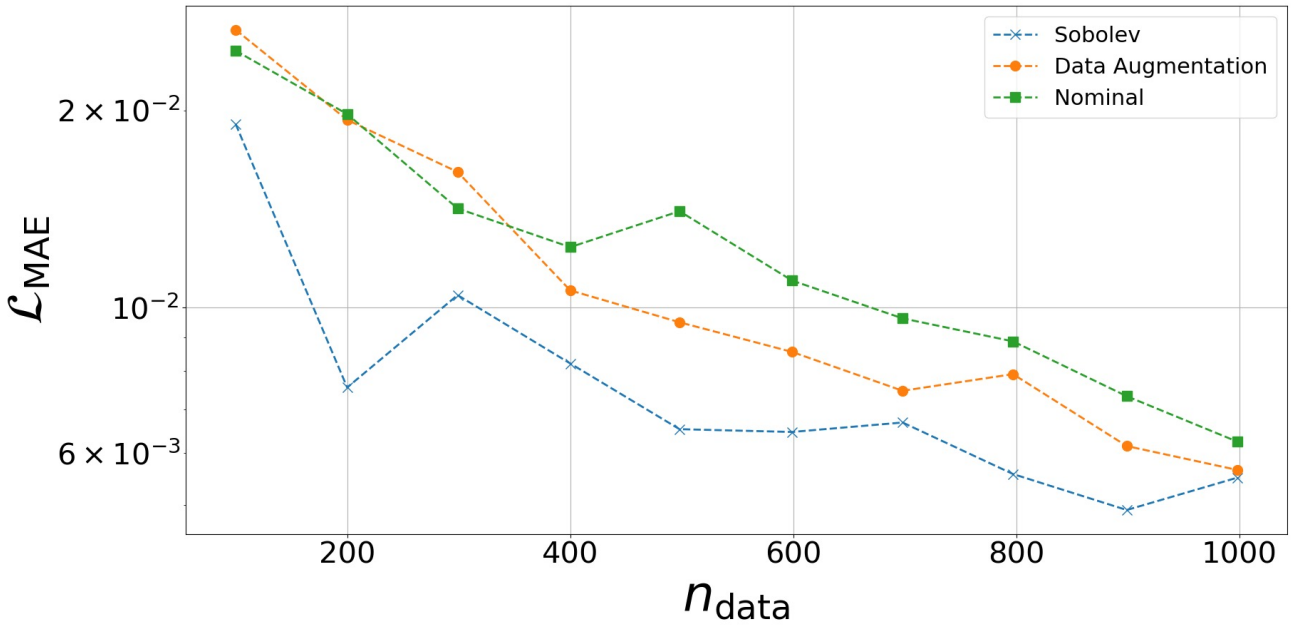


[K. Klatt and S. Engell, 1998, Computers & Chemical Engineering]

Sobolev Training with NLP Sensitivities

All details in [Lüken, Brandner and Lucia, IFAC WC 2023]

- Hyperparameter
  - Grid search on approx. NMPC with nominal training
  - 3 Layers, 100 Neurons
  - Tanh activation
- Evaluation on mean absolute error (MAE)
  - Predictions only: $\|\boldsymbol{u}_0 - \hat{\boldsymbol{u}}_0\|$
  - Less sensitive to outliers



- Sobolev training useful in low data regime
- Large sensitivities might lead to high approximation error

Outline

- Approximate MPC with neural networks
- Efficient sampling via Sobolov Training
- Deterministic and probabilistc guarantees

Can we trust a NN controller?

Safe performance with NN controllers

In general, only an approximate solution is obtained and thus there are **no guarantees** about constraint satisfaction or stability

Many research groups have worked on these issues in recent years

Two main possibilities to achieve guarantees:

- Deterministic guarantees
- Probabilistic guarantees

Deterministic guarantees available mostly for linear systems only

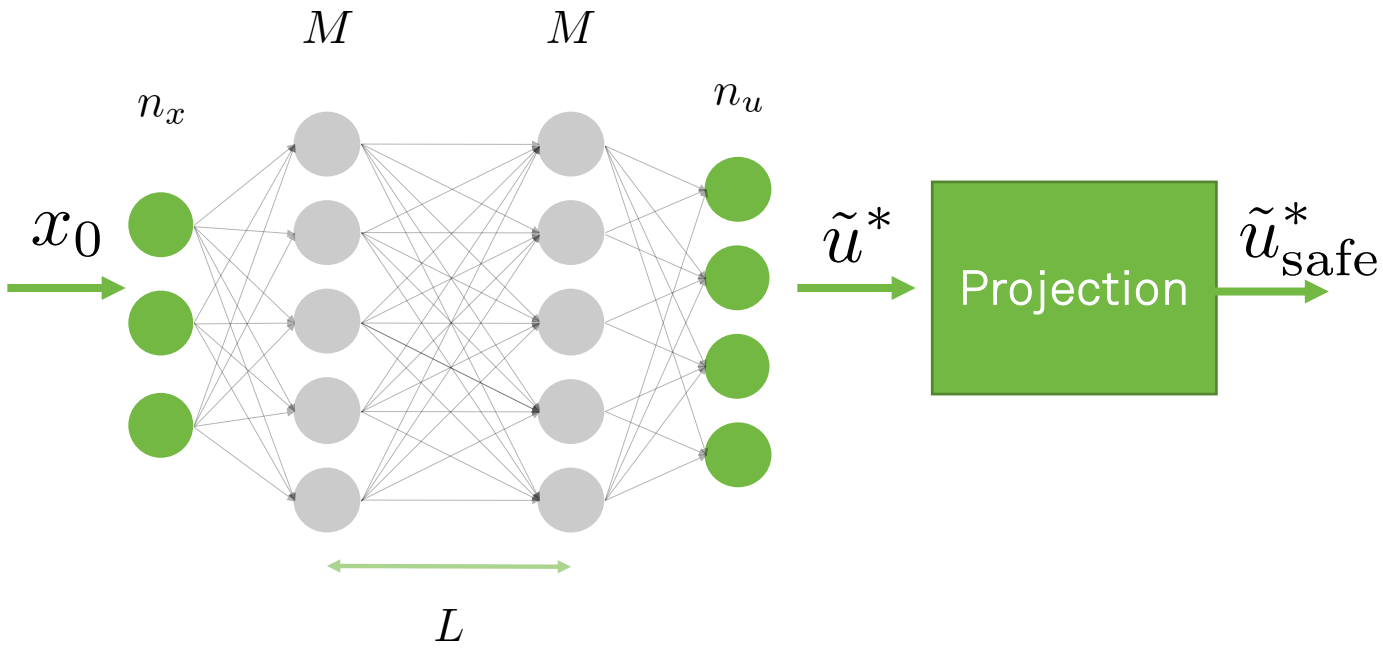
Deterministic guarantees with NN controllers

Linear systems:

- Projection-based methods [Chen et al., 2018, Karg & Lucia, 2020]
- Optimization layers [Maddalena et al., 2020]
- Worst-case approximation errors and Lipschitz constants [Fabiani and Goulart, TAC 2023], [Schwan, Jones and Kuhn, TAC 2023]
- A-priori verification via output-range analysis [Karg & Lucia, 2020]

Deterministic guarantees: Projection

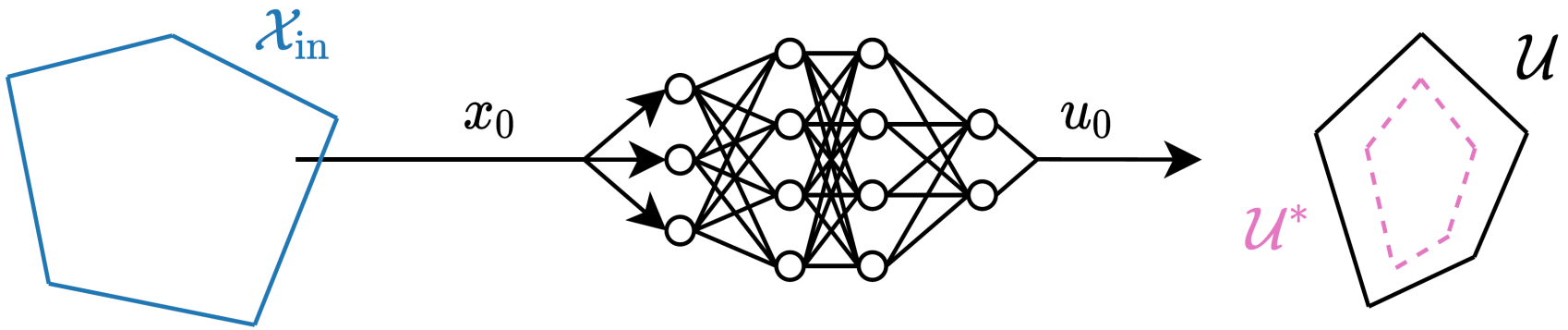
- Projection onto a feasible reagion



$$\begin{aligned} \underset{\hat{u}}{\text{minimize}} \quad & \|\mathcal{N}(x;\theta,M,L) - \hat{u}\|_2^2 \\ \text{subject to} \quad & C_{\text{inv}}(Ax_{\text{init}} + B\hat{u}) \leq c_{\text{inv}}, \\ & C_u\hat{u} \leq c_u, \end{aligned}$$

[Chen et al., ACC 2018, Karg and Lucia IEEE Tran. Cyb. 2020]

Output-range analysis (via MILPs)



- Compute (an over-approximation $\mathcal{U}^*$ of) the output set $\mathcal{U}_{\text{true}} = \mathcal{N}(\mathcal{X})$ of a neural network for a given input set $\mathcal{X}_{\text{in}}$
- Check if $\mathcal{U}^*$ satisfies desired properties, e.g. $\mathcal{U}^* \subseteq \mathcal{U}$

Input constraints

$a_u u_0 \leq b_u$:

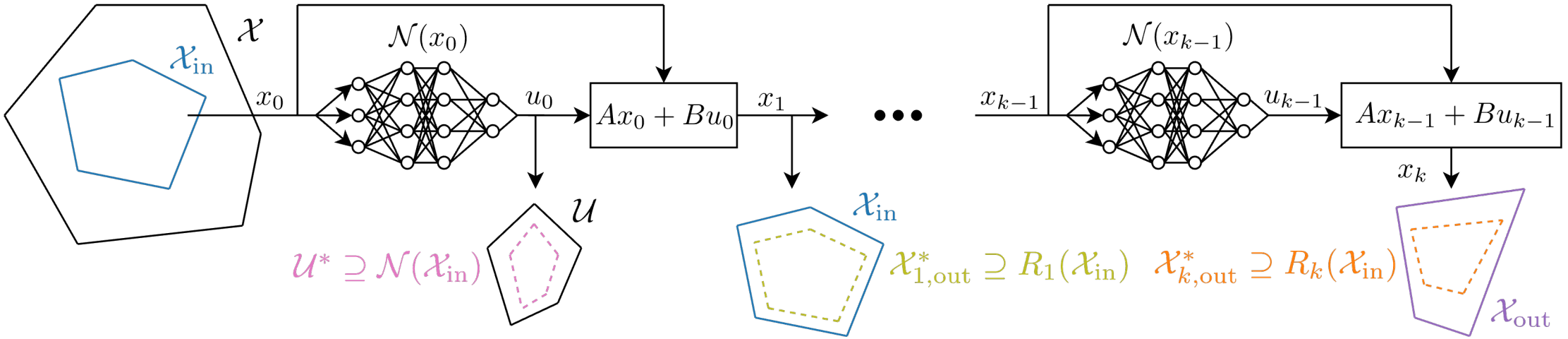
- $z = [z_0, \ldots, z_L]$:
  Output of layers
- $t = [t_1, \ldots, t_L]$:
  Activation hidden layers
- u_0 :
  Largest control input in the direction of the normal of the hyperplane
- x_0 :
  Feasible state leading to largest control input
- $M \in \mathbb{R}$ with $M \geq \max_{l \in [L+1], i \in [n_l]} z_l^{(i)}$

$$
\begin{aligned}
\underset{z,t,u_0,x_0}{\text{maximize}} \quad & a_u u_0 \\
\text{subject to} \quad & C_{\text{in}} x_0 \leq c_{\text{in}}, \\
& z_0 = x_0, \\
& u_0 = W_{L+1} z_L + b_{L+1}, \\
& \text{for all } l \in [L] : \\
& \quad z_l \geq W_l z_{l-1} + b_l, \\
& \quad z_l \leq W_l z_{l-1} + b_l + M t_l, \\
& \quad z_l \geq 0, \\
& \quad z_l \leq M(\mathbf{1} - t_l), \\
& \quad t_l \in \{0,1\}^{n_l},
\end{aligned}
$$

- If $a_u u_0^* \leq b_u$,
  then $\mathcal{N}(x_0) \leq b_u \; \forall x_0 \in \mathcal{X}_{\text{in}}$

[Karg and Lucia, CDC 2020]

Output-range analysis via MILPs



$$\mathcal{R}_k(\mathcal{X}_{\text{in}}) = f_{\text{cl}} \circ \cdots \circ f_{\text{cl}}(\mathcal{X}_{\text{in}}) \text{ with } f_{\text{cl}}(x_k) = Ax_k + B\mathcal{N}(x_k)$$

Control inputs:

$$\mathcal{N}(\mathcal{X}_{\text{in}}) \subseteq \mathcal{U}$$

Control-invariance:

$$\mathcal{R}_1(\mathcal{X}_{\text{in}}) \subseteq \mathcal{X}_{\text{in}}$$

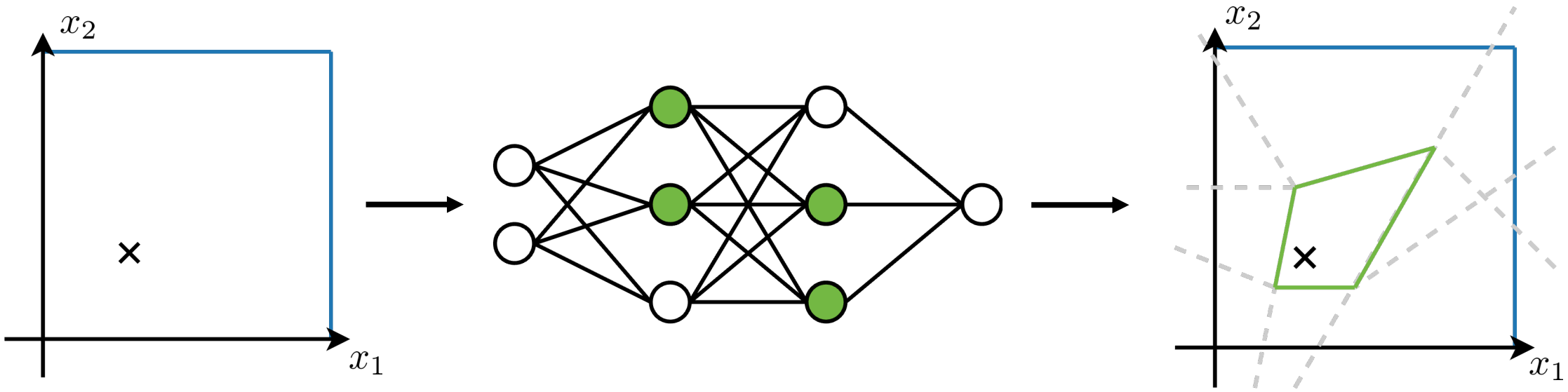
Convergence:

$$\mathcal{R}_k(\mathcal{X}_{\text{in}}) \subseteq \mathcal{X}_{\text{out}}$$

Extended to uncertain systems in [Karg and Lucia, CDC 2022]

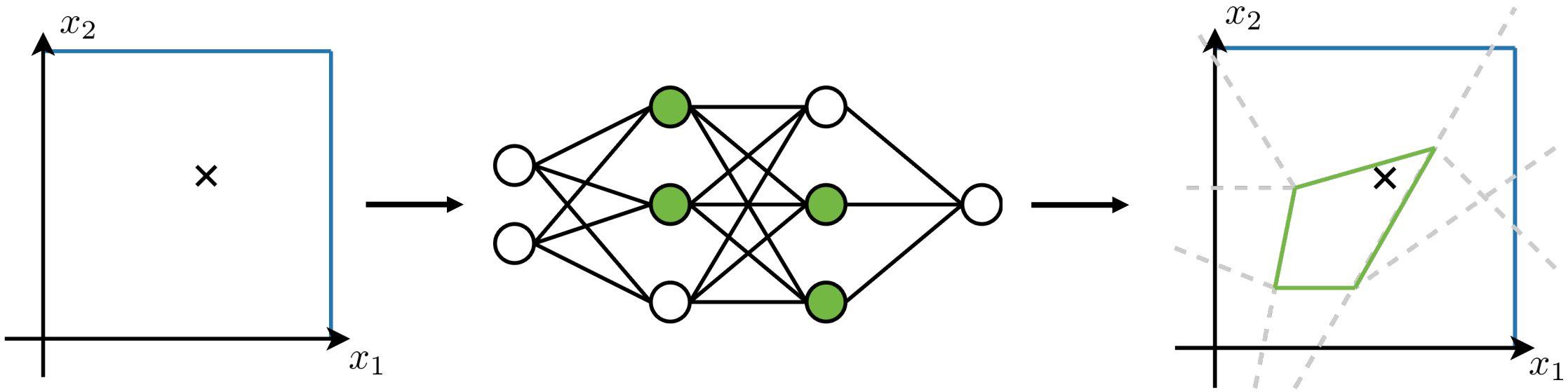
Activation pattern

Each activation pattern defines a region within the state space with an affine state feedback



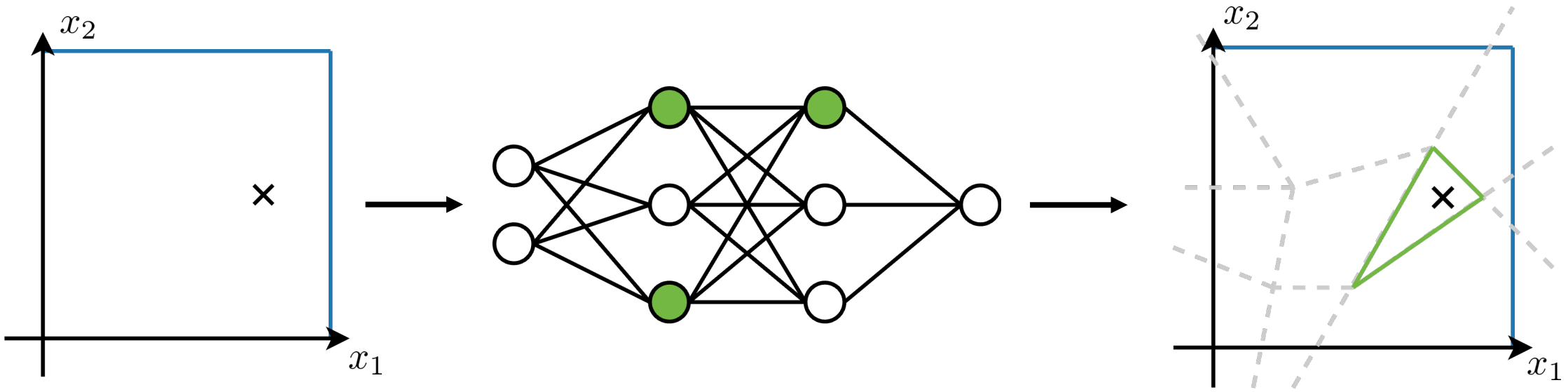
Activation pattern

Each activation pattern defines a region within the state space with an affine state feedback

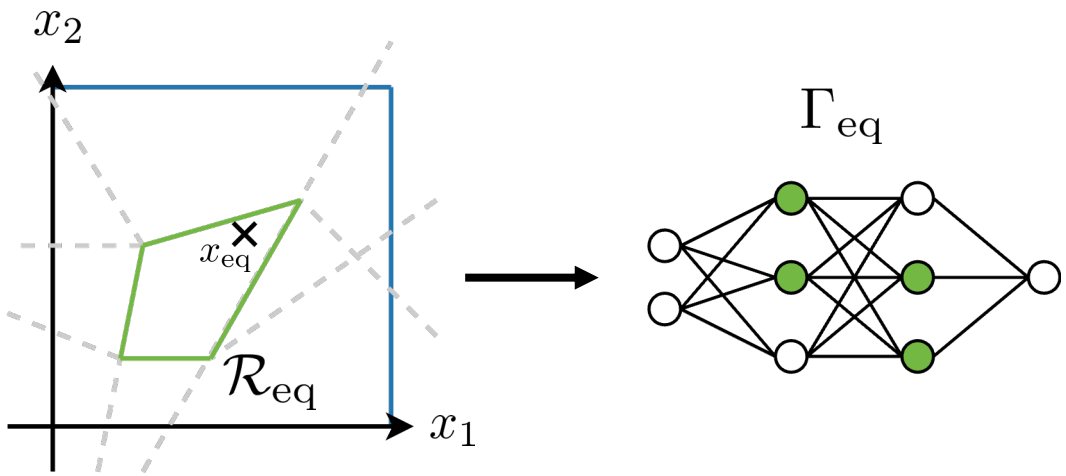


Activation pattern

Each activation pattern defines a region within the state space with an affine state feedback



Feedback in the neighborhood of equilibrium



Feedback: $\mathcal{P}(x, \Gamma_{\text{eq}}, L+1) =$

$$W_{L+1}(W_{\Gamma_{\text{eq}},L}x + b_{\Gamma_{\text{eq}},L}) + b_{L+1} =$$

$$(W_{L+1}W_{\Gamma_{\text{eq}},L})x + (W_{L+1}b_{\Gamma_{\text{eq}},L} + b_{L+1})$$

State feedback

- Feedback:

$$u = -Kx$$

- Stability condition:

$$\|A - BK\| < 1.$$

Feedback in the neighborhood of equilibrium

Neural network

- Feedback:

$$\mathcal{P}(x, \Gamma_{\text{eq}}, L+1) = W_{L+1}(W_{\Gamma_{\text{eq}},L}x + b_{\Gamma_{\text{eq}},L}) + b_{L+1}$$

- Stability condition:

$$W_{L+1}b_{\Gamma_{\text{eq}},L} + b_{L+1} = 0$$
$$\|A + BW_{L+1}W_{\Gamma_{\text{eq}},L}\| < 1$$

State feedback

- Feedback:

$$u = -Kx$$

- Stability condition:

$$\|A - BK\| < 1.$$

▶ Which neighborhood guarantees asymptotic stability?

Asymptotic stability

Theorem 1 [Karg and Lucia, CDC 2020]

If the neural network controller satisfies

$$W_{L+1} b_{\Gamma_{\mathrm{eq}},L} + b_{L+1} = 0,$$
$$\|A + B W_{L+1} W_{\Gamma_{\mathrm{eq}},L}\| < 1,$$

and there exists a $k \in \mathbb{N}$ such that $\mathcal{X}^*_{k,\mathrm{out}} \subseteq \mathcal{R}_{\mathrm{as}}$,

then the closed-loop system $x_{k+1} = f_{\mathrm{cl}}(x_k)$ is asymptotically stable for all $x \in \mathcal{X}_{\mathrm{in}}$.

- In general, neural network controllers do not satisfy these conditions!
- But we can modify the last layer without affecting the shape of the equilibrium region!

LQR-optimal adaptation

Theorem 2 [Karg and Lucia, CDC 2020]

The convex optimization problem

$$\begin{aligned} \underset{\hat{W}_{L+1},\hat{b}_{L+1}}{\text{minimize}} \quad & \sum_{i,j=1}^{n_u,n_L} (\hat{W}_{L+1}^{(i,j)} - W_{L+1}^{(i,j)})^2 + \sum_{i=1}^{n_u} (\hat{b}_{L+1}^{(i)} - b_{L+1}^{(i)})^2 \\ \text{subject to} \quad & \hat{W}_{L+1} W_{\Gamma_{\text{eq}}} = -K_{\text{lqr}}, \\ & \hat{W}_{L+1} b_{\Gamma_{\text{eq}}} + \hat{b}_{L+1} = 0, \end{aligned}$$

admits a solution which provides weights for the last layer of the neural network controller, such that it behaves like an LQR in the neighborhood of the equilibrium.

Oscillating mass

Two states:

- Displacement s
- Velocity v

One control input:

- Force u

System matrices:

$$A = \begin{bmatrix} 0.5403 & -0.8415 \\ 0.8415 & 0.5403 \end{bmatrix}, B = \begin{bmatrix} -0.4597 \\ 0.8415 \end{bmatrix}$$

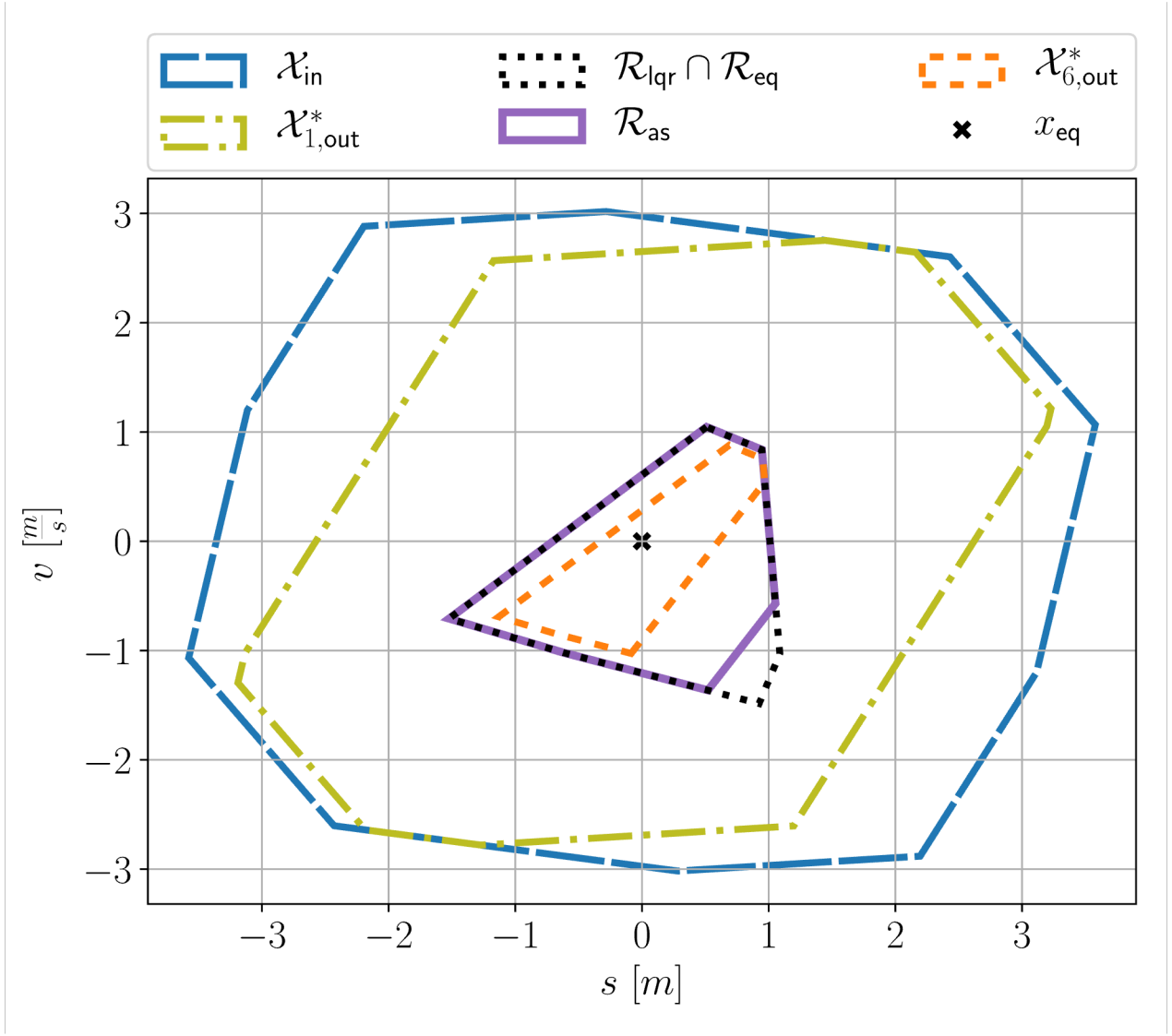
State constraints:

$$\mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid -5 \leq x \leq 5\}$$

Input constraints:

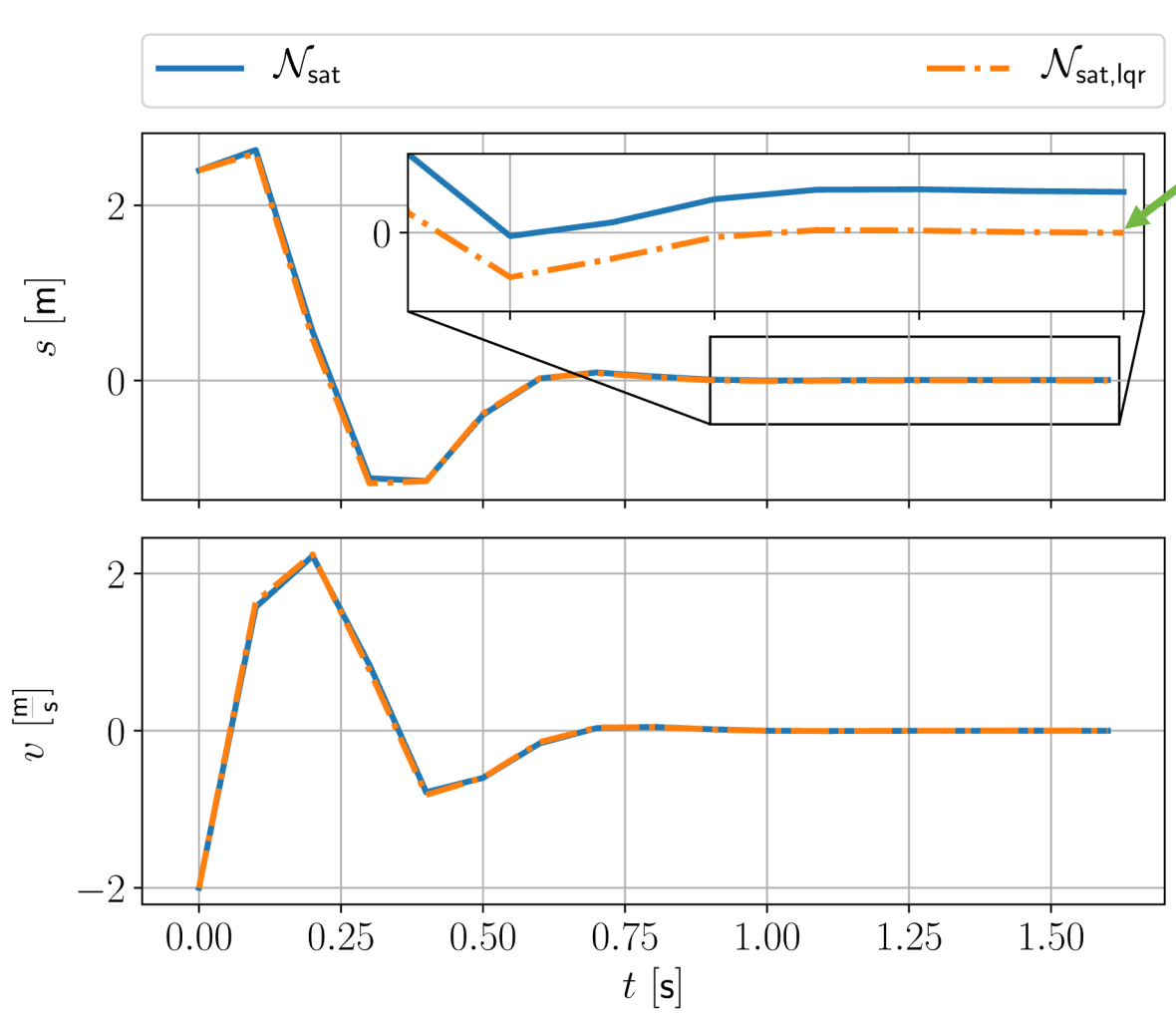
$$\mathcal{U} = \{x \in \mathbb{R}^{n_u} \mid -1 \leq u \leq 1\}$$

Control-invariance and stability



- $\mathcal{X}_{\text{in}}$ is a control-invariant set for neural network controller
- Convergence to $\mathcal{R}_{\text{as}}$ within 6 steps guaranteeing asymptotic stability

Asymptotic stability



Convergence to origin

- LQR-optimized network converges to equilibrium
- Original network has remaining offset error

What if the system is nonlinear and uncertain?

Probabilistic guarantees

- Define any measurable performance function $\phi(w) : \mathcal{W} \to \mathbb{R}$

  For example: $\phi(w) = ||\mathcal{N}(w; \theta, M, L) - \pi_{\mathrm{MPC}}(w)||$

  w represents a bounded uncertainty with known prob. distribution

What is the probability that a certain performance level is achieved?

$$\Pr_{\mathcal{W}}\{\phi(w) \leq \gamma\}$$

Probability estimation

Naive approaches when N i.i.d samples $\phi(w^{(i)})$ are drawn

- Relative frequency

$$\hat{P}_N = \frac{\text{count}(\phi(w^{(i)}) \leq \gamma)}{N}$$

How many samples are needed to obtain a reliable estimate?

- Two-level probability: accuracy ϵ with confidence $1 - \delta$

$$\Pr_{\mathcal{W}}\{|\Pr_{\mathcal{W}}\{\phi(w) \leq \gamma\} - \hat{P}_N| \leq \epsilon\} \geq 1 - \delta$$

Well known bounds... and great improvements

Bernoulli law of large numbers

$$N \geq \frac{1}{4\epsilon^2\delta}$$

Chernoff bound

$$N \geq \frac{\ln(2/\delta)}{2\epsilon^2}$$

- Independent of the number of uncertainties and distribution
- But it often leads to large number of samples

$$\epsilon = \delta = 0.005 \longrightarrow N = 119,830$$

Well known bounds... and great improvements

Fortunately, Chernoff bound can be improved! [Tempo, Bai, Dabbene, 1997]

- Focus on the worst-case performance

$$\hat{\phi}_N = \max_{i=1,\dots,N} \phi(w^{(i)})$$

If the number of samples is such that $N \geq \frac{1}{\epsilon} \ln \frac{1}{\delta}$ [Hertneck et al., L-CSS 2018]
then the following inequality holds:

$$\Pr_{\mathcal{W}}\{\Pr_{\mathcal{W}}\{\phi(w) > \hat{\phi}_N\} \leq \epsilon\} \geq 1 - \delta$$

use this method to verify optimality using the dual [Zhang et al., ACC 2019]

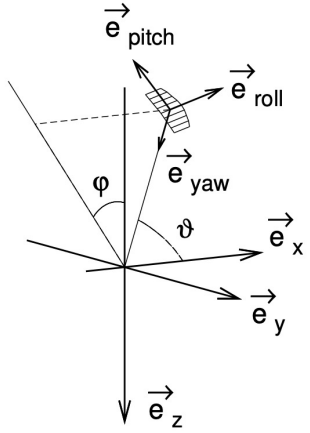
$$\epsilon = \delta = 0.005 \quad \longrightarrow \quad N \geq 1,058$$

Discard the r worst cases to improve performance [Alamo et al., 2018]

A towing kite

Probabilistically safe, embedded robust output-feedback NMPC

$$\dot{\theta}_{\text{kite}} = \frac{v_{\text{a}}}{L_{\text{T}}}(\cos\psi_{\text{kite}} - \frac{\tan\theta_{\text{kite}}}{E}),$$
$$\dot{\phi}_{\text{kite}} = -\frac{v_{\text{a}}}{L_{\text{T}}\sin\theta_{\text{kite}}}\sin\psi_{\text{kite}},$$
$$\dot{\psi}_{\text{kite}} = \frac{v_{\text{a}}}{L_{\text{T}}}\tilde{u} + \dot{\phi}_{\text{kite}}\cos\theta_{\text{kite}},$$



Erhard and Strauch, 2012

- Objective is to maximize thrust
- Two states can be measured, EKF to estimate
- Uncertain aerodynamic coefficients and wind parameters
- Minimum height constraint

[Karg, Alamo and Lucia, *Int. Journal of Robust and Nonlinear Control*, 2021]

Probabilistic validation

Define the performance function (with backoff η)

$$\phi(w; N_{\text{sim}}, \kappa_{\text{dnn},\eta}) = \max_{i=0,\ldots,N_{\text{sim}}} (h_{\min} - h(x(j, w))),$$

The controller is probabilistically safe if with probability $1 - \delta$:

$$\Pr_{\mathcal{W}}(\phi(w; N_{\text{sim}}, \kappa_{\text{dnn},\eta}) > 0) \leq \epsilon,$$

We need the following amount of samples:

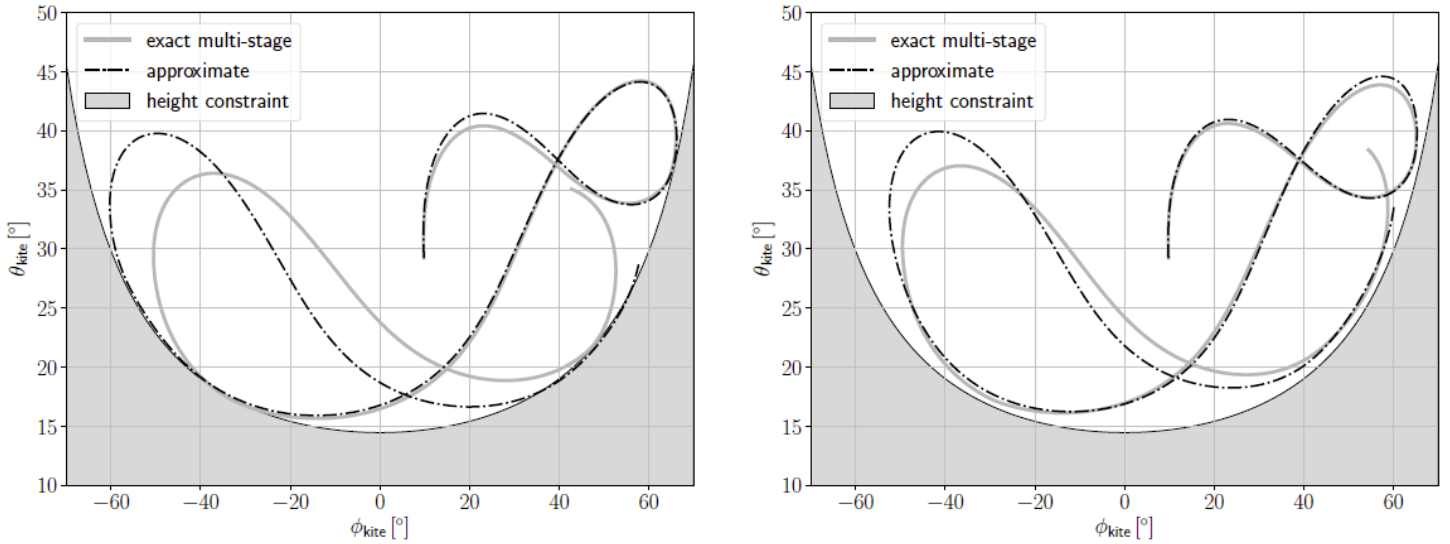
$$\epsilon = 0.02, \delta = 1 \times 10^{-6}, r = 4 \quad \longrightarrow \quad N = 1388$$

[Karg, Alamo and Lucia, *Int. Journal of Robust and Nonlinear Control*, 2021]

Results

Embedded real-time implementation on an ARM-Cortex M3

- 96 kB memory footprint, 32 ms running time for DNN and 28 ms for EKF



(a) $\eta = 0\,\text{m}$ **(b)** $\eta = 4\,\text{m}$

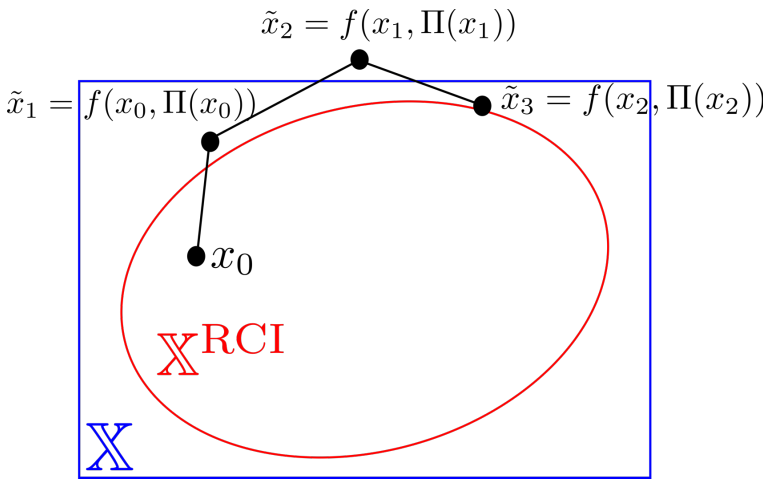
controller	$\kappa_{\text{dnn},0}$	$\kappa_{\text{dnn},2}$	$\kappa_{\text{dnn},4}$	$\kappa_{\text{dnn},6}$
feasible trajectories	660/1388	1380/1388	1385/1388	1387/1388
$\psi(\mathbf{v}, 4)$ [m]	1.682	0.273	-0.316	-1.818
T_{F} (avg.) [kN]	227.516	225.997	224.185	222.179
probabilistically safe	No	No	Yes	Yes

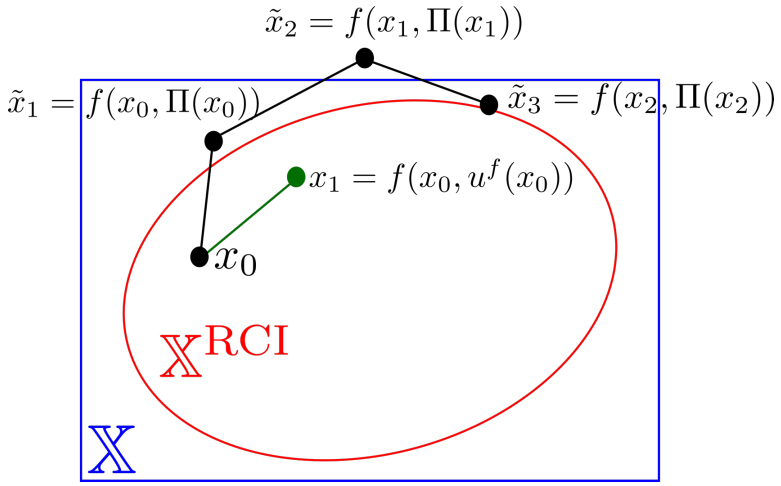
What about deterministic guarantees for nonlinear uncertain systems?

Deterministic case: nonlinear & uncertain

In general a very difficult problem. One possibility:

- Online multi-step predictions of the closed loop [Hose et al., arXiv:2304.09575, 2023]
  1. Simulate online whether the closed-loop leads to a safe set
  2. If it does, apply approximate MPC
  3. If not, apply fallback controller





- How to compute a fallback controller? How to check safety for uncertain systems? Exploit system properties! (In this case, monotonicity)
  [Coogan, CDC 2020, Heinlein et al., CDC 2022, Adamek et al. L-CSS 2024]

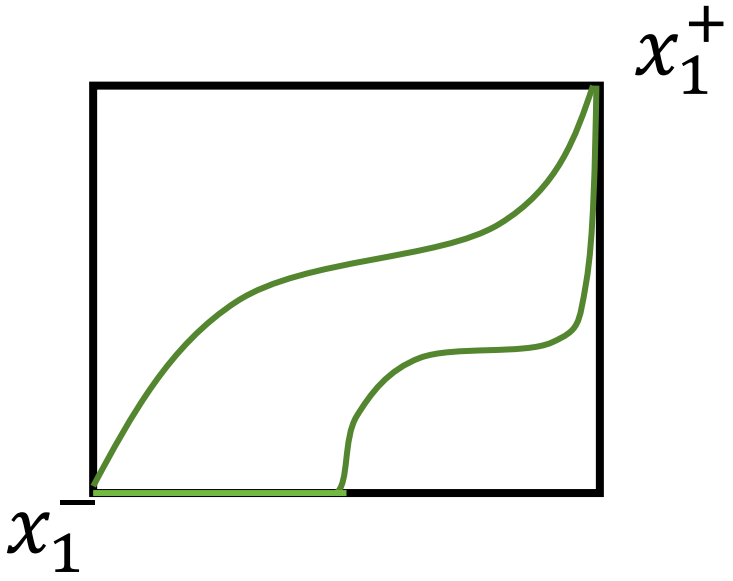
Monotone systems – states and uncertainties

- Here only discrete-time systems: $x^+ = f(x, u, p)$
- For $x_1 \geq x_2, p_1 \geq p_2$
  - $f(x_1, u, p) \geq f(x_2, u, p), \forall\, p \in P, \forall\, u \in U$
  - $f(x, u, p_1) \geq f(x, u, p_2), \forall\, x \in X, \forall\, u \in U$
- Present in many engineering fields, e.g.
  - Temperature control in buildings [1]
  - Biochemical reaction casacades [2]
  - Traffic control [3]
- Exploited in control settings (reachability analysis) [4]

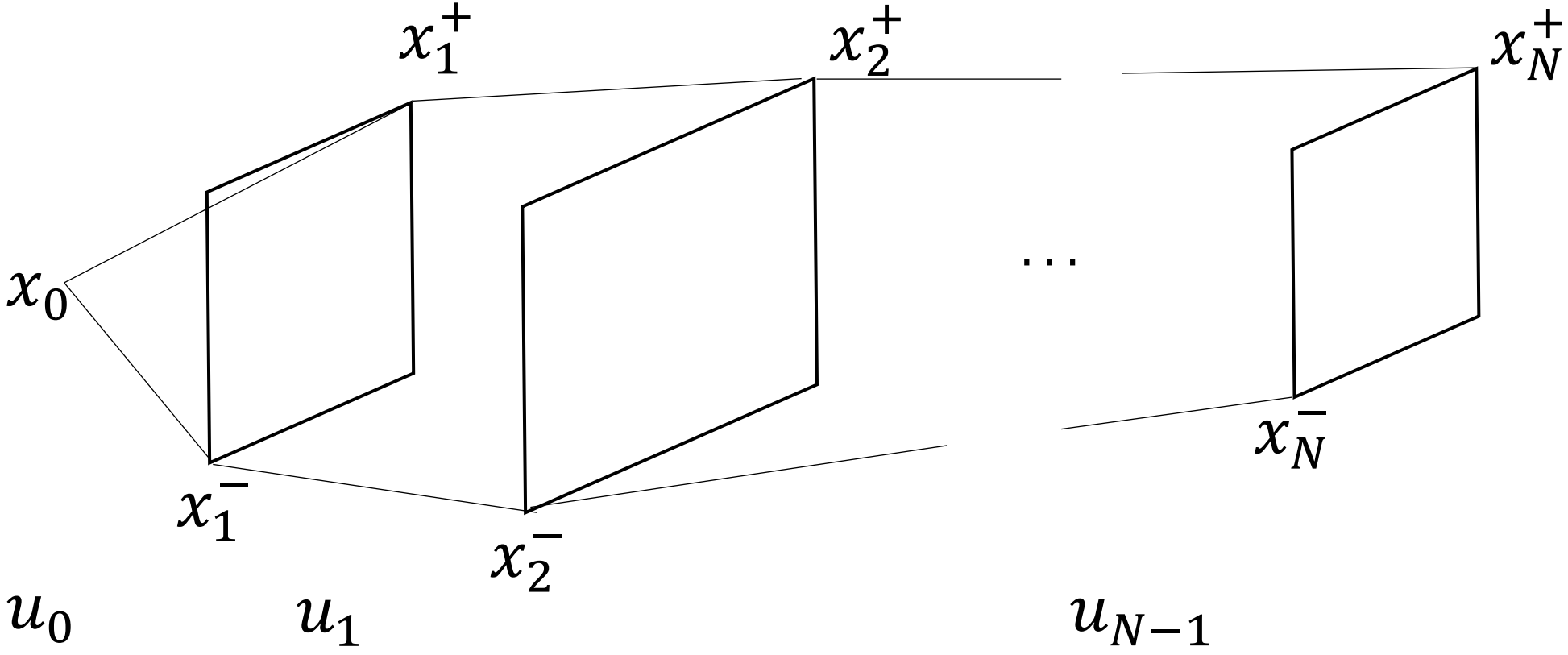
[1] P.-J. Meyer, A. Girard, and E. Witrant, "*Automatica*,, 2016.
[2] C. Kallies, M. Schliemann, R. Findeisen, S. Lucia, and E. Bullinger, *IFAC-PapersOnLine*, 2016.
[3] M. Schmitt, C. Ramesh, P. Goulart, and J. Lygeros, ACC 2017
[4] S. Coogan, CDC 2022

Reachable sets for monotone systems

- This allows an easy computation of reachable sets for bounded uncertainties $p \in [p^-, p^+]$
- $x_1 \in [f(x_0, u_0, p^-), f(x_0, u_0, p^+)] = [x_1^-, x_1^+]$
- The reachable sets can be propagated
- $x_2 \in [f(x_1^-, u_1, p^-), f(x_1^+, u_1, p^+)] = [x_2^-, x_2^+]$



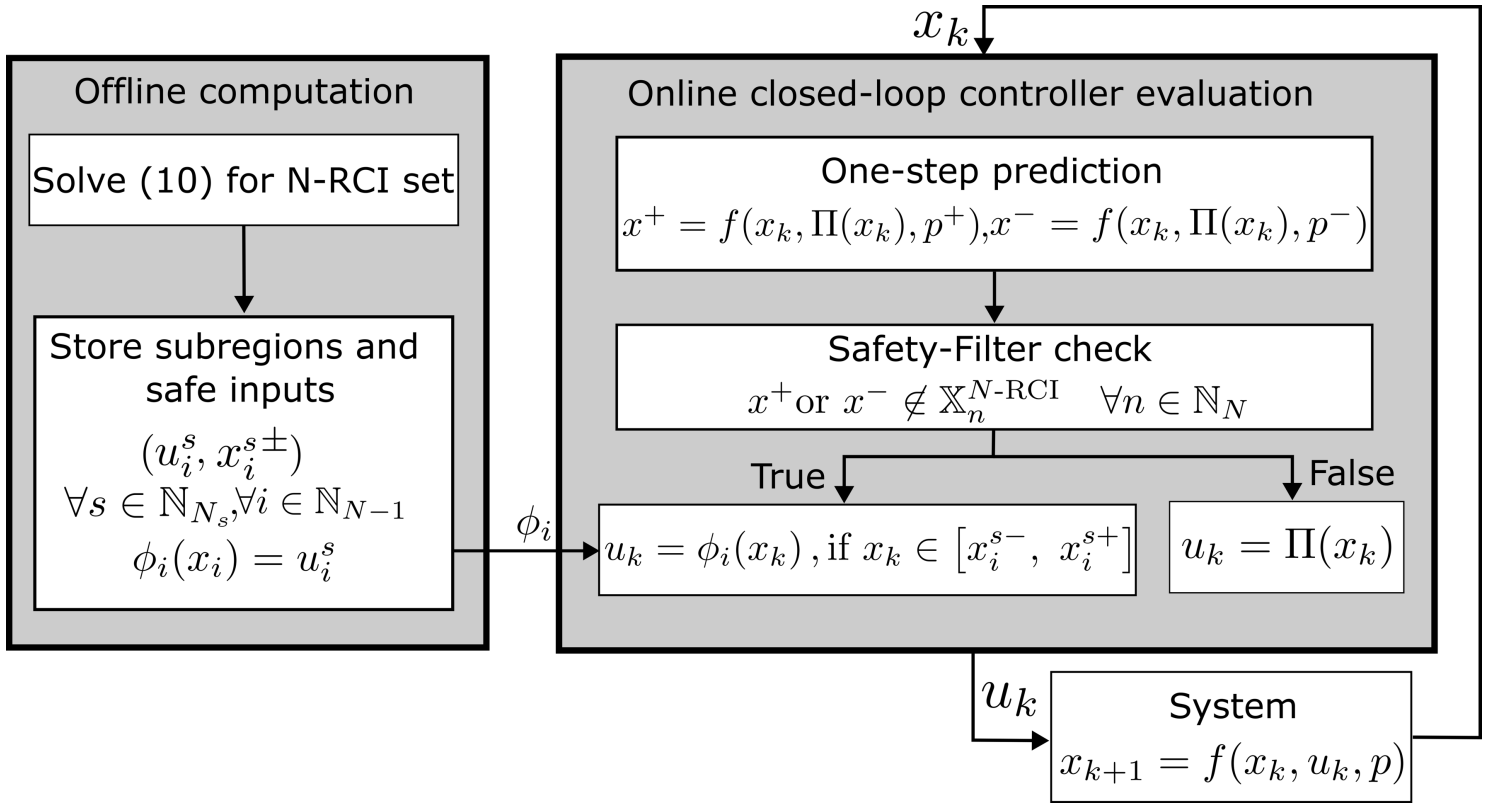
Multiple step reachable sets



- Reachable sets over the whole prediction horizon

Details in [Heinlein et al., CDC 2022] and [Adamek, Heinlein, Lüken & Lucia, L-CSS 2024]

NN controller with deterministic guarantees



Details in [Adamek, Heinlein, Lüken & Lucia, L-CSS 2024]

Conclusions

- Approximate MPC via NNs is a powerful approach that can enable (robust) (N)MPC virtually everywhere
- Guarantees are possible!
  - Deterministic:
    - Many possibilitie in the linear case, often resulting in MILPs / MIQPs
    - Difficult in the nonlinear case: assume fallback strategies or exploit system properties
  - Probabilistic:
    - Easier for a posterior validation
- Exploit the possibilities of fast closed-loop simulations

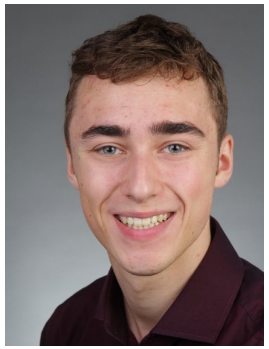
Open problems

- Approximate MPC via NNs is nice, but how does it scale?
- Worst-case errors of approximate MPC are sometimes large
- What about distribution shifts in probabilistic validation?
- Can we have a constructive approach that obtains guarantees for the nonlinear uncertain case?

Acknowledgements

- Thanks to the great work of my PhD Students


Benjamin Karg


Moritz Heinlein


Joshua Adamek


Lukas Lüken


Dean Brandner

- Collaborators: **Teo Alamo**, Moritz Diehl, Ali Mesbah, Joel Paulson
- And funding agencies:

Funded by

DFG Deutsche Forschungsgemeinschaft

German Research Foundation

This is the end