Computing control Lyapunov functions with neural networks

Lars Grüne

Mathematical Institute, University of Bayreuth, Germany

based on joint work with Mario Sperl (Bayreuth) and Debasish Chatterjee (Mumbai)

supported by DFG Deutsche Forschungsgemeinschaft

InterCoML
Control Theory & Machine Learning

Kickoff Meeting of WG2, 5 December 2025

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(x(t), u(t)), \quad x(0) = x_0,$$

or in discrete time

$$x^+(t) := x(t+1) = g(x(t), u(t)), \quad x(0) = x_0,$$

where $f, g \colon \mathbb{R}^n \times U \to \mathbb{R}^n$ is a controlled vector field or map, respectively

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(x(t), u(t)), \quad x(0) = x_0,$$

or in discrete time

$$x^+(t) := x(t+1) = g(x(t), u(t)), \quad x(0) = x_0,$$

where $f, g\colon \mathbb{R}^n \times U \to \mathbb{R}^n$ is a controlled vector field or map, respectively

Objective:

- control the state towards a desired set or set point and keep it there

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(x(t), u(t)), \quad x(0) = x_0,$$

or in discrete time

$$x^+(t) := x(t+1) = g(x(t), u(t)), \quad x(0) = x_0,$$

where $f, g \colon \mathbb{R}^n \times U \to \mathbb{R}^n$ is a controlled vector field or map, respectively

Objective:

- control the state towards a desired set or set point and keep it there
- design the control in feedback form, i.e., $u(t) = F(x(t))$

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(x(t), u(t)), \quad x(0) = x_0,$$

or in discrete time

$$x^+(t) := x(t+1) = g(x(t), u(t)), \quad x(0) = x_0,$$

where $f, g\colon \mathbb{R}^n \times U \to \mathbb{R}^n$ is a controlled vector field or map, respectively

Objective: Feedback stabilisation

- control the state towards a desired set or set point and keep it there
- design the control in feedback form, i.e., $u(t) = F(x(t))$

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(x(t), u(t)), \quad x(0) = x_0,$$

or in discrete time

$$x^+(t) := x(t+1) = g(x(t), u(t)), \quad x(0) = x_0,$$

where $f, g\colon \mathbb{R}^n \times U \to \mathbb{R}^n$ is a controlled vector field or map, respectively

Objective: Feedback stabilisation

- control the state towards a desired set or set point and keep it there
- design the control in feedback form, i.e., $u(t) = F(x(t))$

The feedback allows to react to deviations of the real world system from its mathematical model used for designing the control

Setting

We consider nonlinear control systems in continuous time

$$\dot{x}(t) := \frac{d}{dt}x(t) = f(x(t), u(t)), \quad x(0) = x_0,$$

or in discrete time

$$x^+(t) := x(t+1) = g(x(t), u(t)), \quad x(0) = x_0,$$

where $f, g\colon \mathbb{R}^n \times U \to \mathbb{R}^n$ is a controlled vector field or map, respectively

Objective: Feedback stabilisation

- control the state towards a desired set or set point and keep it there
- design the control in feedback form, i.e., $u(t) = F(x(t))$

The feedback allows to react to deviations of the real world system from its mathematical model used for designing the control (of course, F must be designed properly such that the control reacts reasonably to such deviations)

Lyapunov functions

Lyapunov functions

We first explain Lyapunov functions without control and in continuous time

We consider autonomous ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0$$

with $f : \mathbb{R}^n \to \mathbb{R}^n$

Lyapunov functions

We first explain Lyapunov functions without control and in continuous time

We consider autonomous ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0$$

with $f : \mathbb{R}^n \to \mathbb{R}^n$

Assume $x^* = 0$ is an equilibrium, i.e., $f(0) = 0$

Lyapunov functions

We first explain Lyapunov functions without control and in continuous time

We consider autonomous ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0$$

with $f : \mathbb{R}^n \to \mathbb{R}^n$

Assume $x^* = 0$ is an equilibrium, i.e., $f(0) = 0$

A continuously differentiable $V : \mathbb{R}^n \to \mathbb{R}_0^+$ is a Lyapunov function, if there are functions $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{K}_\infty$ such that

$$\begin{aligned} \alpha_1(\|x\|) \ \leq \ V(x) \ &\leq \ \alpha_2(\|x\|) \\ DV(x)f(x) \ &\leq \ -\alpha_3(\|x\|) \end{aligned}$$

UNIVERSITÄT BAYREUTH

Lyapunov functions

We first explain Lyapunov functions without control and in continuous time

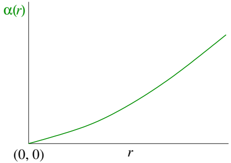
We consider autonomous ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0$$

with $f : \mathbb{R}^n \to \mathbb{R}^n$

Assume $x^* = 0$ is an equilibrium, i.e., $f(0) = 0$

$\alpha \in \mathcal{K}_\infty$: $\alpha : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, continuous, strictly increasing, $\alpha(0) = 0$, unbounded



Lyapunov functions

We first explain Lyapunov functions without control and in continuous time

We consider autonomous ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0$$

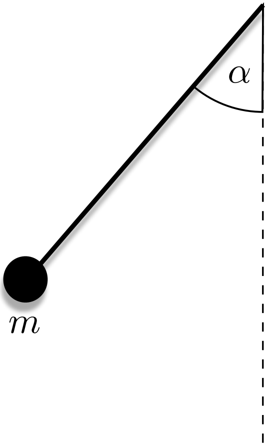
with $f : \mathbb{R}^n \to \mathbb{R}^n$

Assume $x^* = 0$ is an equilibrium, i.e., $f(0) = 0$

A continuously differentiable $V : \mathbb{R}^n \to \mathbb{R}_0^+$ is a Lyapunov function, if there are functions $\alpha_1, \alpha_2, \alpha_3 \in \mathcal{K}_\infty$ such that

$$\begin{aligned} \alpha_1(\|x\|) \;\; &\leq \;\; V(x) \;\; \leq \;\; \alpha_2(\|x\|) \\ DV(x)f(x) \;\; &\leq \;\; -\alpha_3(\|x\|) \end{aligned}$$

Example: Mathematical Pendulum



$x_1 = \alpha =$ angle

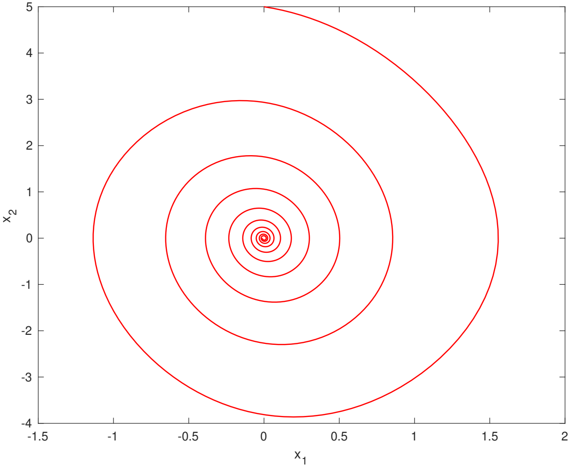
$x_2 =$ angular velocity

$\rightsquigarrow$ ordinary differential equation

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -g\sin(x_1) - \frac{k}{m}x_2$$

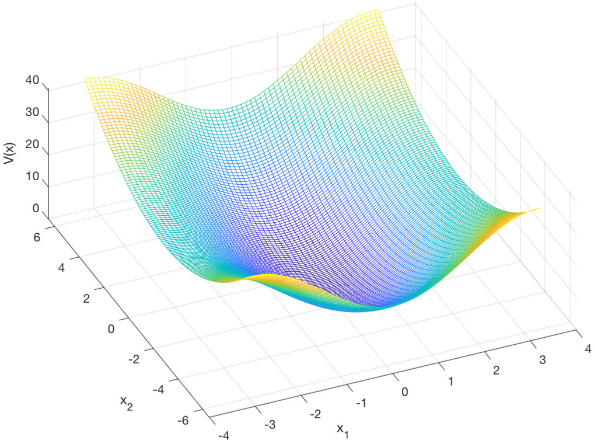
Pendulum solution and Lyapunov function



Solution of pendulum equation

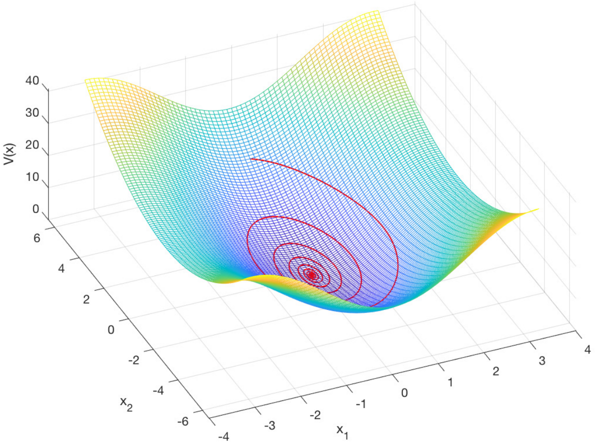
UNIVERSITÄT BAYREUTH

Pendulum solution and Lyapunov function



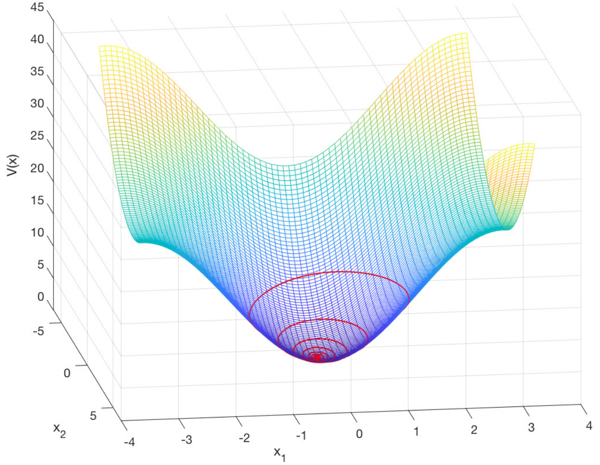
Lyapunov function $V(x) = x_2^2/2 + g(1 - \cos x_1) + 0.1 x_2 \sin(x_1)$

Pendulum solution and Lyapunov function



Lyapunov function with solution superimposed

Pendulum solution and Lyapunov function



Lyapunov function with solution superimposed

Control Lyapunov functions

A smooth control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \ \le \ \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

Control Lyapunov functions

A smooth control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \;\le\; \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

- A control u approximately realizing the "inf" yields a solution along which V decreases

Control Lyapunov functions

A smooth control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \le \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

- A control u approximately realizing the "inf" yields a solution along which V decreases
- The (approximately) minimising u only depends on x and is thus in feedback form

UNIVERSITÄT BAYREUTH

Control Lyapunov functions

A smooth control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|)$$

$$\inf_{u \in U} DV(x)f(x,u) \leq -\alpha_3(\|x\|)$$

- A control u approximately realizing the “inf” yields a solution along which V decreases
- The (approximately) minimising u only depends on x and is thus in feedback form
- Depending on the regularity of V, robustness of the feedback w.r.t. model errors and disturbances can be proved

Control Lyapunov functions

A smooth control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \ \le \ \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

- A control u approximately realizing the "inf" yields a solution along which V decreases
- The (approximately) minimising u only depends on x and is thus in feedback form
- Depending on the regularity of V, robustness of the feedback w.r.t. model errors and disturbances can be proved

A clf hence acts like a road map, showing the way to the desired set or equilibrium

Control Lyapunov functions

Problem: A smooth control Lyapunov function (clf)

$$\alpha_1(\|x\|) \le V(x) \;\le\; \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|) \qquad (*)$$

may not exist

Control Lyapunov functions

Problem: A smooth control Lyapunov function (clf)

$$\alpha_1(\|x\|) \le V(x) \ \le \ \alpha_2(\|x\|)$$
$$\inf_{u \in U} DV(x) f(x,u) \le -\alpha_3(\|x\|) \qquad (*)$$

may not exist

Remedies:

- Replace $DV(x)f(x,u)$ in $(*)$ by the Dini derivative

UNIVERSITÄT BAYREUTH

Control Lyapunov functions

Problem: A smooth control Lyapunov function (clf)

$$\begin{aligned} \alpha_1(\|x\|) \le V(x) &\le \alpha_2(\|x\|) \\ \inf_{u\in U} DV(x)f(x,u) &\le -\alpha_3(\|x\|) \end{aligned} \qquad (*)$$

may not exist

Remedies:

- Replace $DV(x)f(x,u)$ in $(*)$ by the Dini derivative
- Switch to discrete time and replace $(*)$ by

$$\inf_{u\in U} V(g(x,u)) \le V(x) - \alpha_3(\|x\|)$$

Numerical computation of Lyapunov functions

Various numerical approaches for computing (control) Lyapunov functions have been developed over the years:

- Series expansion [Kirin et al. '82]
- Semi-Lagrangian schemes [Camilli/Gr./Wirth '00, Falcone/Gr./Wirth '00]
- Finite elements and linear programming [Hafstein '02ff]
- Sum-of-squares methods [Papachristodoulou/Prajna '02]
- Radial Basis functions [Giesl '04ff, Giesl/Wendland '07ff]

Numerical computation of Lyapunov functions

Various numerical approaches for computing (control) Lyapunov functions have been developed over the years:

- Series expansion [Kirin et al. '82]
- Semi-Lagrangian schemes [Camilli/Gr./Wirth '00, Falcone/Gr./Wirth '00]
- Finite elements and linear programming [Hafstein '02ff]
- Sum-of-squares methods [Papachristodoulou/Prajna '02]
- Radial Basis functions [Giesl '04ff, Giesl/Wendland '07ff]

All these methods suffer from the curse of dimensionality: The numerical effort grows exponentially with the space dimension n

Numerical computation of Lyapunov functions

Various numerical approaches for computing (control) Lyapunov functions have been developed over the years:

- Series expansion [Kirin et al. '82]
- Semi-Lagrangian schemes [Camilli/Gr./Wirth '00, Falcone/Gr./Wirth '00]
- Finite elements and linear programming [Hafstein '02ff]
- Sum-of-squares methods [Papachristodoulou/Prajna '02]
- Radial Basis functions [Giesl '04ff, Giesl/Wendland '07ff]

All these methods suffer from the curse of dimensionality: The numerical effort grows exponentially with the space dimension n

Deep neural networks as approximation architecture are believed to mitigate this effort

Deep neural networks

Neural network approaches for Lyapunov functions

The computation of control Lyapunov functions via neural networks has been investigated since more than 30 years:

[Sontag '92, Long/Bayoumi '93, Abu-Khalaf/Lewis '04, Serpen '05, Petridis/Petridis '06, Noroozi/Karimaghaee/Safaei/Javadi '08, Khansari-Zadeh/Billard '14, Richards/Berkenkamp/Krause '18]

Neural network approaches for Lyapunov functions

The computation of control Lyapunov functions via neural networks has been investigated since more than 30 years:

[Sontag '92, Long/Bayoumi '93, Abu-Khalaf/Lewis '04, Serpen '05, Petridis/Petridis '06, Noroozi/Karimaghaee/Safaei/Javadi '08, Khansari-Zadeh/Billard '14, Richards/Berkenkamp/Krause '18]

These early papers investigate representability, different kinds of parametrizations, and different ways of implementing $DV(x)f(x,u)$

Neural network approaches for Lyapunov functions

The computation of control Lyapunov functions via neural networks has been investigated since more than 30 years:

[Sontag '92, Long/Bayoumi '93, Abu-Khalaf/Lewis '04, Serpen '05, Petridis/Petridis '06, Noroozi/Karimaghaee/Safaei/Javadi '08, Khansari-Zadeh/Billard '14, Richards/Berkenkamp/Krause '18]

These early papers investigate representability, different kinds of parametrizations, and different ways of implementing $DV(x)f(x,u)$ (they are not obsolete!)

Neural network approaches for Lyapunov functions

The computation of control Lyapunov functions via neural networks has been investigated since more than 30 years:

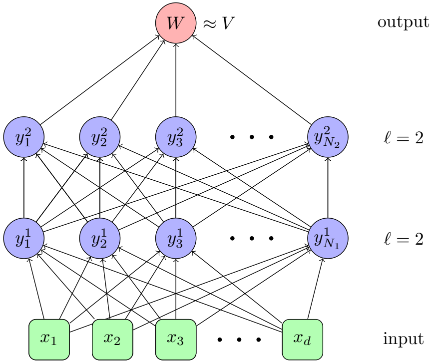
[Sontag '92, Long/Bayoumi '93, Abu-Khalaf/Lewis '04, Serpen '05, Petridis/Petridis '06, Noroozi/Karimaghaee/Safaei/Javadi '08, Khansari-Zadeh/Billard '14, Richards/Berkenkamp/Krause '18]

These early papers investigate representability, different kinds of parametrizations, and different ways of implementing $DV(x)f(x,u)$ (they are not obsolete!)

In what follows, I will present four recent developments in this field:

- Mitigating the curse of dimensionality
- Avoiding singularities
- Verification
- Nonsmooth clfs

Deep neural network with 2 hidden layers



$$W_\theta(x) = a \cdot y^2 + c$$

$$y^2_k = \sigma^2(w^2_k \cdot y^1 + b^2_k)$$

$$y^1_k = \sigma^1(w^1_k \cdot x + b^1_k)$$

w^1_k, w^2_k, a = vectors of weights, "$\cdot$" = scalar product

b^1_k, b^2_k, c = scalar parameters, $\sigma^1, \sigma^2 : \mathbb{R} \to \mathbb{R}$ = activation functions

Examples: $\sigma(r) = r$, $\sigma(r) = \max\{r, 0\}$, $\sigma(r) = \ln(e^r + 1)$, $\sigma(r) = \frac{1}{1+e^{-r}}$

θ = vector of all parameters $(w^\ell_k, b^\ell_k, a, c)$

$W_{\theta^*}(x) \approx V(x)$, approximated Lyapunov function for "trained" θ^*

UNIVERSITÄT BAYREUTH

Training and minimizing over u

Training is usually done similar to learning PDE or ODE solutions with PINNs:

We minimize $\sum_i L(W\theta(x_i), D_x W_\theta(x_i), x_i)$ w.r.t. θ at sampling points x_i

Training and minimizing over u

Training is usually done similar to learning PDE or ODE solutions with PINNs:

We minimize $\sum_i L(W\theta(x_i), D_x W_\theta(x_i), x_i)$ w.r.t. θ at sampling points x_i,

where the loss function L penalizes the violation of the inequalities that define the clf, e.g.,

$$L(w,p,x) = \left[\min_{u\in U} pf(x,u) + \alpha_3(\|x\|)\right]_+^2 + \nu\left([w-\alpha_1(\|x\|)]_-^2 + [w-\alpha_2(\|x\|)]_+^2\right)$$

with $[a]_- := \min\{a,0\}$, $[a]_+ := \max\{a,0\}$, and weight $\nu > 0$

Training and minimizing over u

Training is usually done similar to learning PDE or ODE solutions with PINNs:

We minimize $\sum_i L(W\theta(x_i), D_x W_\theta(x_i), x_i)$ w.r.t. θ at sampling points x_i,

where the loss function L penalizes the violation of the inequalities that define the clf, e.g.,

$$L(w,p,x) = \left[\min_{u\in U} pf(x,u) + \alpha_3(\|x\|)\right]_+^2 + \nu\left([w-\alpha_1(\|x\|)]_-^2 + [w-\alpha_2(\|x\|)]_+^2 \right)$$

with $[a]_- := \min\{a,0\}$, $[a]_+ := \max\{a,0\}$, and weight $\nu > 0$

Note: clfs can be characterized by optimal control, but this may be inefficient when using neural networks

UNIVERSITÄT BAYREUTH

Training and minimizing over u

Training is usually done similar to learning PDE or ODE solutions with PINNs:

We minimize $\sum_i L(W\theta(x_i), D_x W_\theta(x_i), x_i)$ w.r.t. θ at sampling points x_i,

where the loss function L penalizes the violation of the inequalities that define the clf, e.g.,

$$L(w,p,x) = \left[\min_{u\in U} pf(x,u) + \alpha_3(\|x\|)\right]_+^2 + \nu\left([w-\alpha_1(\|x\|)]_-^2 + [w-\alpha_2(\|x\|)]_+^2\right)$$

with $[a]_- := \min\{a,0\}$, $[a]_+ := \max\{a,0\}$, and weight $\nu > 0$

Note: clfs can be characterized by optimal control, but this may be inefficient when using neural networks

Challenge: How to compute $\min_{u\in U} pf(x,u)$ inside the minimization of L?

Training and minimizing over u

Training is usually done similar to learning PDE or ODE solutions with PINNs:

We minimize $\sum_i L(W\theta(x_i), D_x W_\theta(x_i), x_i)$ w.r.t. θ at sampling points x_i,

where the loss function L penalizes the violation of the inequalities that define the clf, e.g.,

$$L(w,p,x) = \left[\min_{u\in U} pf(x,u) + \alpha_3(\|x\|)\right]_+^2 + \nu\left([w-\alpha_1(\|x\|)]_-^2 + [w-\alpha_2(\|x\|)]_+^2\right)$$

with $[a]_- := \min\{a,0\}$, $[a]_+ := \max\{a,0\}$, and weight $\nu > 0$

Note: clfs can be characterized by optimal control, but this may be inefficient when using neural networks

Challenge: How to compute $\min_{u\in U} pf(x,u)$ inside the minimization of L?

One possibility: If $f(x,u) = f_1(x) + f_2(x)u$ and $u \in [-C, C]^m$, then

$$\min_{u\in U} pf(x,u) = pf_1(x) - C\|pf_2(x)\|_1$$

UNIVERSITÄT BAYREUTH

Mitigating the curse of dimensionality

Universal approximation theorem

Known: Every continuous function can be approximated arbitrarily good by a DNN ("Universal approximation theorem" [Cybenko '89, Mhaskar '96])

Universal approximation theorem

Known: Every continuous function can be approximated arbitrarily good by a DNN ("Universal approximation theorem" [Cybenko '89, Mhaskar '96]) — but the number of neurons needed grows exponentially with the dimensions n

Universal approximation theorem

Known: Every continuous function can be approximated arbitrarily good by a DNN ("Universal approximation theorem" [Cybenko '89, Mhaskar '96]) — but the number of neurons needed grows exponentially with the dimensions n

$\rightsquigarrow$ Curse of dimensionality applies also here

Universal approximation theorem

Known: Every continuous function can be approximated arbitrarily good by a DNN ("Universal approximation theorem" [Cybenko '89, Mhaskar '96]) — but the number of neurons needed grows exponentially with the dimensions n

$\rightsquigarrow$ Curse of dimensionality applies also here

But: For functions with beneficial structures this approximation works with only polynomial effort

- Functions with a high degree of smoothness and suitable form of the Fourier transformation (e.g., Barron functions)

Universal approximation theorem

Known: Every continuous function can be approximated arbitrarily good by a DNN ("Universal approximation theorem" [Cybenko '89, Mhaskar '96]) — but the number of neurons needed grows exponentially with the dimensions n

$\rightsquigarrow$ Curse of dimensionality applies also here

But: For functions with beneficial structures this approximation works with only polynomial effort

- Functions with a high degree of smoothness and suitable form of the Fourier transformation (e.g., Barron functions)

  These were exploited for 2nd order HJB equations by Darbon, E, Han, Hutzenthaler, Jentzen, Kruse, and others

Universal approximation theorem

Known: Every continuous function can be approximated arbitrarily good by a DNN ("Universal approximation theorem" [Cybenko '89, Mhaskar '96]) — but the number of neurons needed grows exponentially with the dimensions n

$\rightsquigarrow$ Curse of dimensionality applies also here

But: For functions with beneficial structures this approximation works with only polynomial effort

- Functions with a high degree of smoothness and suitable form of the Fourier transformation (e.g., Barron functions)

  These were exploited for 2nd order HJB equations by Darbon, E, Han, Hutzenthaler, Jentzen, Kruse, and others

  $\rightsquigarrow$ Unlikely to work for deterministic problems

UNIVERSITÄT BAYREUTH

Universal approximation theorem

Known: Every continuous function can be approximated arbitrarily good by a DNN (“Universal approximation theorem” [Cybenko ’89, Mhaskar ’96]) — but the number of neurons needed grows exponentially with the dimensions n

$\rightsquigarrow$ Curse of dimensionality applies also here

But: For functions with beneficial structures this approximation works with only polynomial effort

- Functions with a high degree of smoothness and suitable form of the Fourier transformation (e.g., Barron functions)

  These were exploited for 2nd order HJB equations by Darbon, E, Han, Hutzenthaler, Jentzen, Kruse, and others

  $\rightsquigarrow$ Unlikely to work for deterministic problems

- For this reason, here we use separable functions

What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$

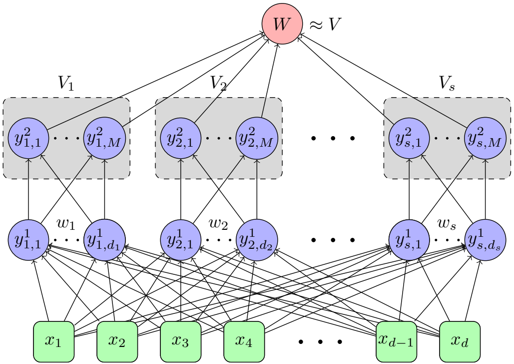
UNIVERSITÄT BAYREUTH

What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$

We approximate the individual V_j by the grey blocks, whose number grows linearly with the dimension d

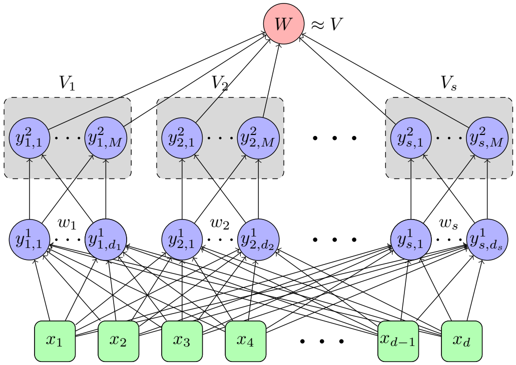


UNIVERSITÄT BAYREUTH

What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$



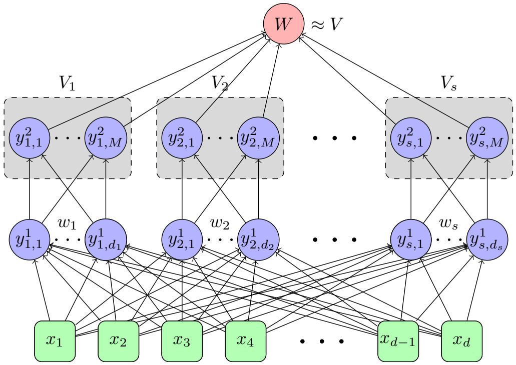
We approximate the individual V_j by the grey blocks, whose number grows linearly with the dimension d

Applying the universal approximation theorem separately in each grey block, we can prove:

What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$



We approximate the individual V_j by the grey blocks, whose number grows linearly with the dimension d

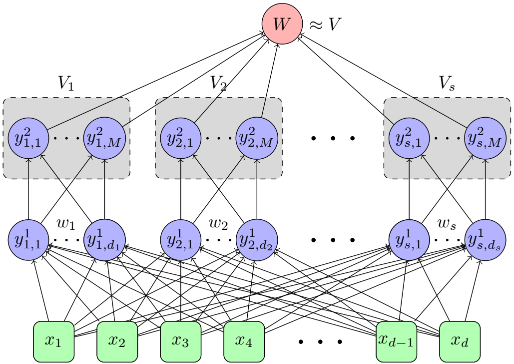
Applying the universal approximation theorem separately in each grey block, we can prove:

Theorem: Functions $V(x) = \sum_{j=1}^{s} V_j(w_j)$ and their derivatives, with $V_j \in \mathcal{W}_1^{d_j}$ and $d_j \leq d_{\max}$ independent of d can be approximated on K with any accuracy $\varepsilon > 0$ with a number of neurons growing only polynomially in d

What are separable functions and why are they beneficial?

Separable function:

$$V(x) = \sum_{j=1}^{s} V_j(w_j), \quad w_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$$



We approximate the individual V_j by the grey blocks, whose number grows linearly with the dimension d

Applying the universal approximation theorem separately in each grey block, we can prove:

Theorem: Functions $V(x) = \sum_{j=1}^{s} V_j(w_j)$ and their derivatives, with $V_j \in \mathcal{W}_1^{d_j}$ and $d_j \leq d_{\max}$ independent of d can be approximated on K with any accuracy $\varepsilon > 0$ with a number of neurons growing only polynomially in d

More precisely, the number of required neurons is $\mathcal{O}\left(\varepsilon^{-d_{\max}}\right) \mathcal{O}\left(d^{d_{\max}+1}\right)$

Nonlinear small-gain theory

For Lyapunov functions, nonlinear small gain theory provides existence results for separable Lyapunov functions

UNIVERSITÄT BAYREUTH

Nonlinear small-gain theory

For Lyapunov functions, nonlinear small gain theory provides existence results for separable Lyapunov functions

It assumes that the system can be decomposed into subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}), \quad i = 1, \ldots, s, \qquad z_{-i} = (z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_s)$$

where the interconnection structure is expressed by a directed graph

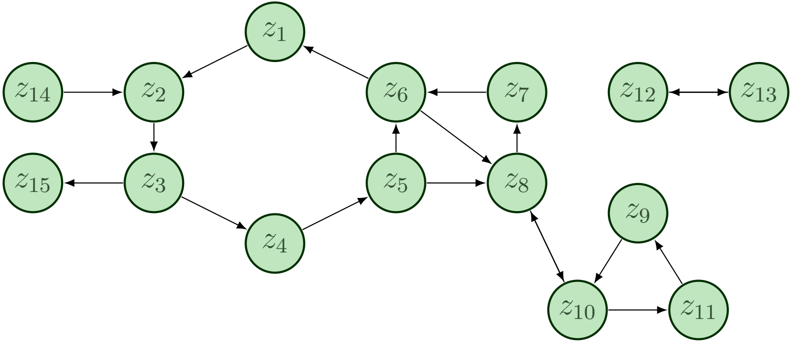
Nonlinear small-gain theory

For Lyapunov functions, nonlinear small gain theory provides existence results for separable Lyapunov functions

It assumes that the system can be decomposed into subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}), \quad i = 1, \dots, s, \qquad z_{-i} = (z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_s)$$

where the interconnection structure is expressed by a directed graph



Nonlinear small-gain theory

For Lyapunov functions, nonlinear small gain theory provides existence results for separable Lyapunov functions

It assumes that the system can be decomposed into subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}), \quad i = 1, \dots, s, \qquad z_{-i} = (z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_s)$$

where the interconnection structure is expressed by a directed graph

The influence of subsystem i on subsystem j is expressed by a gain function γ_{ij}

UNIVERSITÄT BAYREUTH

Nonlinear small-gain theory

For Lyapunov functions, nonlinear small gain theory provides existence results for separable Lyapunov functions

It assumes that the system can be decomposed into subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}), \quad i = 1, \dots, s, \qquad z_{-i} = (z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_s)$$

where the interconnection structure is expressed by a directed graph

The influence of subsystem i on subsystem j is expressed by a gain function γ_{ij}

If in each cycle in the graph the concatenation of the γ_{ij} satisfies

$$\gamma_{i_1 i_2} \circ \gamma_{i_2 i_3} \circ \dots \circ \gamma_{i_m i_1} < \mathrm{id}$$

Nonlinear small-gain theory

For Lyapunov functions, nonlinear small gain theory provides existence results for separable Lyapunov functions

It assumes that the system can be decomposed into subsystems

$$\dot{z}_i = f_i(z_i, z_{-i}), \quad i = 1, \dots, s, \qquad z_{-i} = (z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_s)$$

where the interconnection structure is expressed by a directed graph

The influence of subsystem i on subsystem j is expressed by a gain function γ_{ij}

If in each cycle in the graph the concatenation of the γ_{ij} satisfies

$$\gamma_{i_1 i_2} \circ \gamma_{i_2 i_3} \circ \dots \circ \gamma_{i_m i_1} < \mathrm{id},$$

then a separable Lyapunov function $V(x) = \sum_{j=1}^{s} V_j(z_j)$ exists

[Dashkovskiy/Rüffer/Wirth '10, Dashkovskiy/Ito/Wirth '11]

See also [Jiang/Teel/Praly '94, Jiang/Mareels/Wang '96, Rüffer '07ff, ...]

UNIVERSITÄT BAYREUTH

Control Lyapunov functions

If we assume smoothness, a control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \ \le \ \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

UNIVERSITÄT BAYREUTH

Control Lyapunov functions

If we assume smoothness, a control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \ \le \ \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

Question: When can we employ small gain techniques here?

UNIVERSITÄT BAYREUTH

Control Lyapunov functions

If we assume smoothness, a control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \ \le \ \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

Question: When can we employ small gain techniques here?

Recall the sufficient condition

$$\gamma_{i_1 i_2} \circ \gamma_{i_2 i_3} \circ \ldots \circ \gamma_{i_m i_1} < \mathrm{id} \qquad \text{for each cycle in the graph}$$

Control Lyapunov functions

If we assume smoothness, a control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \;\le\; \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

Question: When can we employ small gain techniques here?

Recall the sufficient condition

$$\gamma_{i_1 i_2} \circ \gamma_{i_2 i_3} \circ \ldots \circ \gamma_{i_m i_1} < \mathrm{id} \qquad \text{for each cycle in the graph}$$

This implies: If in each cycle of the graph there is at least one subsystem for which the γ_{ij} can be made arbitrarily "flat" ("active nodes"), then there exists a clf of the separable form $V(x) = \sum_{j=1}^{s} V_j(z_j)$ [Chen/Astolfi '24]

Control Lyapunov functions

If we assume smoothness, a control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \leq V(x) \;\leq\; \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \leq -\alpha_3(\|x\|)$$

Question: When can we employ small gain techniques here?

Recall the sufficient condition

$$\gamma_{i_1 i_2} \circ \gamma_{i_2 i_3} \circ \ldots \circ \gamma_{i_m i_1} < \mathrm{id} \qquad \text{for each cycle in the graph}$$

This implies: If in each cycle of the graph there is at least one subsystem for which the γ_{ij} can be made arbitrarily "flat" ("active nodes"), then there exists a clf of the separable form $V(x) = \sum_{j=1}^{s} V_j(z_j)$ [Chen/Astolfi '24]

Note: This theory guarantees the existence of a separable clf, but for using this result with DNNs, neither the z_j nor the V_j need to be known in advance

UNIVERSITÄT BAYREUTH

Control Lyapunov functions

If we assume smoothness, a control Lyapunov function (clf) is characterised by

$$\alpha_1(\|x\|) \le V(x) \ \le \ \alpha_2(\|x\|)$$

$$\inf_{u\in U} DV(x)f(x,u) \le -\alpha_3(\|x\|)$$

Question: When can we employ small gain techniques here?

Recall the sufficient condition

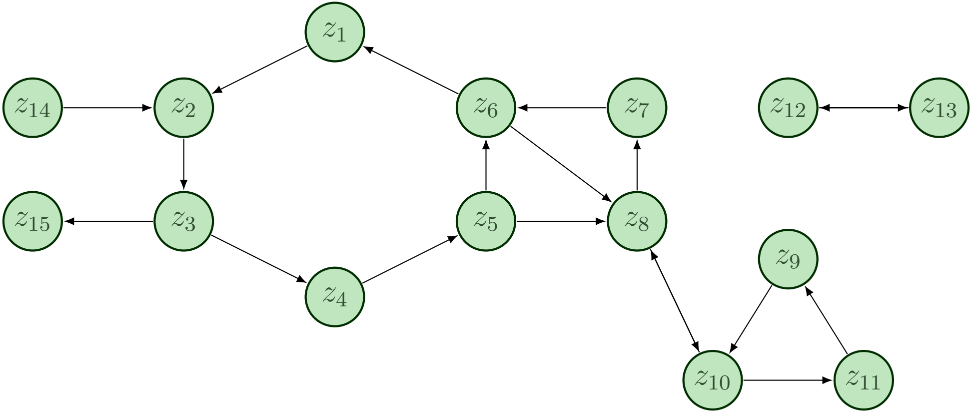
$$\gamma_{i_1 i_2} \circ \gamma_{i_2 i_3} \circ \ldots \circ \gamma_{i_m i_1} < \mathrm{id} \qquad \text{for each cycle in the graph}$$

This implies: If in each cycle of the graph there is at least one subsystem for which the γ_{ij} can be made arbitrarily "flat" ("active nodes"), then there exists a clf of the separable form $V(x) = \sum_{j=1}^{s} V_j(z_j)$ [Chen/Astolfi '24]

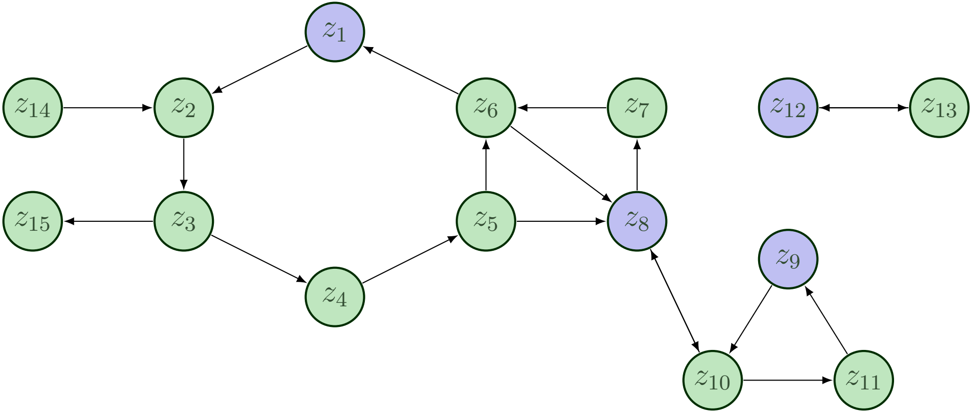
Note: This theory guarantees the existence of a separable clf, but for using this result with DNNs, neither the z_j nor the V_j need to be known in advance

Rather, the network will "learn" this structure during the training process

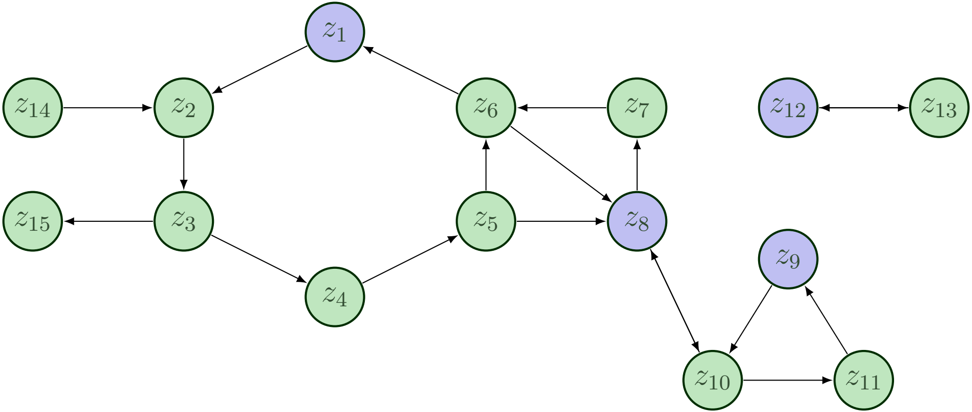
Example for a suitable graph structure



Example for a suitable graph structure



Example for a suitable graph structure

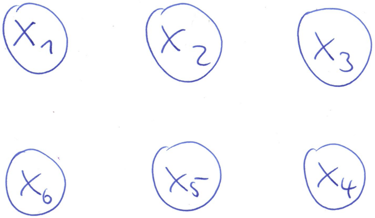


If the blue nodes are active $\rightsquigarrow$ $V(x) = \sum_{j=1}^{15} V_j(z_j)$ exists

Computation with DNN

Example:

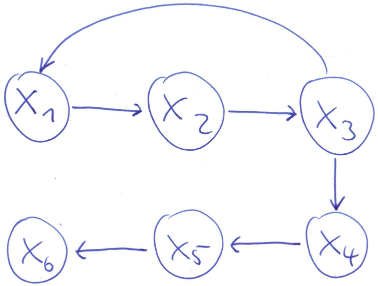
$$
\begin{aligned}
\dot{x}_1 &= x_3 + u \\
\dot{x}_2 &= x_1 - x_2 + x_1^2 \\
\dot{x}_3 &= x_2 - x_3 \\
\dot{x}_4 &= x_3 - x_4 \\
\dot{x}_5 &= x_4 - x_5 \\
\dot{x}_6 &= x_5 - x_6
\end{aligned}
$$



Computation with DNN

Example:

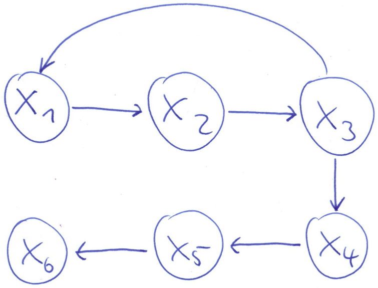
$$
\begin{aligned}
\dot{x}_1 &= x_3 + u \\
\dot{x}_2 &= x_1 - x_2 + x_1^2 \\
\dot{x}_3 &= x_2 - x_3 \\
\dot{x}_4 &= x_3 - x_4 \\
\dot{x}_5 &= x_4 - x_5 \\
\dot{x}_6 &= x_5 - x_6
\end{aligned}
$$



Computation with DNN

Example:

$$
\begin{aligned}
\dot{x}_1 &= x_3 + u \\
\dot{x}_2 &= x_1 - x_2 + x_1^2 \\
\dot{x}_3 &= x_2 - x_3 \\
\dot{x}_4 &= x_3 - x_4 \\
\dot{x}_5 &= x_4 - x_5 \\
\dot{x}_6 &= x_5 - x_6
\end{aligned}
$$



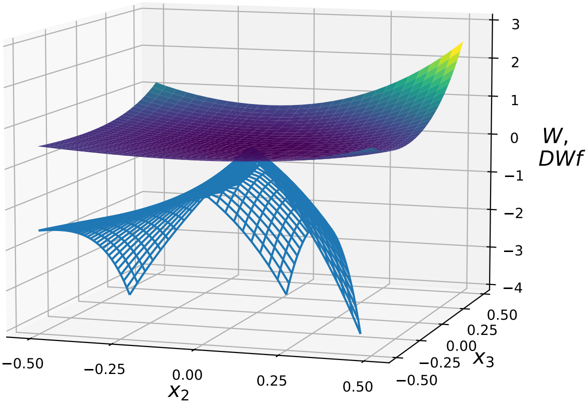
$\rightsquigarrow$ $$V(x) = \sum_{j=1}^{6} V(x_j)$$

Computation with DNN

Example:

$$\begin{aligned}
\dot{x}_1 &= x_3 + u \\
\dot{x}_2 &= x_1 - x_2 + x_1^2 \\
\dot{x}_3 &= x_2 - x_3 \\
\dot{x}_4 &= x_3 - x_4 \\
\dot{x}_5 &= x_4 - x_5 \\
\dot{x}_6 &= x_5 - x_6
\end{aligned}$$

$\rightsquigarrow$ $$V(x) = \sum_{j=1}^{6} V(x_j)$$

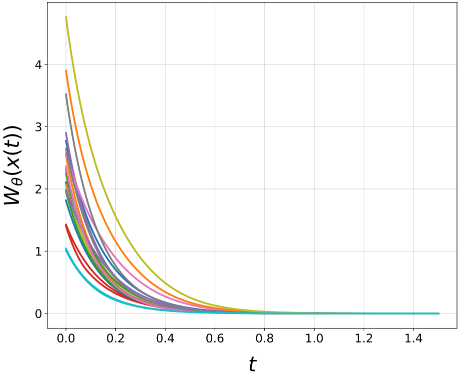


Computation time: 820s

Computation with DNN

Example [Ahmadi/Krstic/Parrilo '11]:

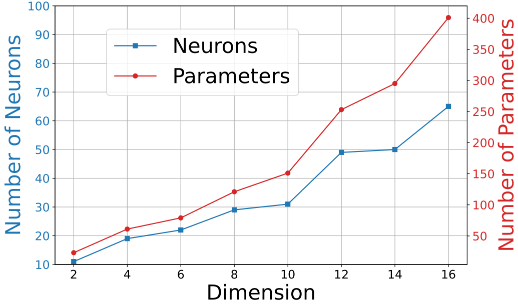
$$\dot{x} = \begin{pmatrix} -x_1 + x_1x_2 - 0.1x_9^2 \\ -x_2u_1 \\ -x_3 + x_3x_4 - 0.1x_1^2 \\ -x_4u_2 \\ -x_5 + x_5x_6 + 0.1x_7^2 \\ -x_6u_3 \\ -x_7 + x_7x_8 \\ -x_8u_4 \\ -x_9 + x_9x_{10} \\ -x_{10}u_5 + 0.1x_2^2 \end{pmatrix}$$



Computation with DNN

Example [Ahmadi/Krstic/Parrilo '11]:

$$\dot{x} = \begin{pmatrix} -x_1 + x_1 x_2 - 0.1 x_9^2 \\ -x_2 u_1 \\ -x_3 + x_3 x_4 - 0.1 x_1^2 \\ -x_4 u_2 \\ -x_5 + x_5 x_6 + 0.1 x_7^2 \\ -x_6 u_3 \\ -x_7 + x_7 x_8 \\ -x_8 u_4 \\ -x_9 + x_9 x_{10} \\ -x_{10} u_5 + 0.1 x_2^2 \end{pmatrix}$$



Avoiding singularities

The singularity problem

An approximate (control) Lyapunov function W_{θ^*} will only satisfy the inequalities

$$\alpha_1(\|x\|) \le W_{\theta^*}(x) \le \alpha_2(\|x\|)$$

$$\inf_{u\in U} DW_{\theta^*}(x)f(x,u) \le -\alpha_3(\|x\|)$$

approximately

UNIVERSITÄT BAYREUTH

The singularity problem

An approximate (control) Lyapunov function W_{θ^*} will only satisfy the inequalities

$$-\varepsilon + \alpha_1(\|x\|) \le W_{\theta^*}(x) \le \alpha_2(\|x\|) + \varepsilon$$

$$\inf_{u \in U} DW_{\theta^*}(x) f(x,u) \le -\alpha_3(\|x\|) + \varepsilon$$

UNIVERSITÄT BAYREUTH

The singularity problem

An approximate (control) Lyapunov function W_{θ^*} will only satisfy the inequalities

$$-\varepsilon + \alpha_1(\|x\|) \le W_{\theta^*}(x) \le \alpha_2(\|x\|) + \varepsilon$$

$$\inf_{u\in U} DW_{\theta^*}(x)f(x,u) \le -\alpha_3(\|x\|) + \varepsilon$$

Since $\alpha_i(\|x\|) \approx 0$ for $x \approx 0$, this means that W_{θ^*} and $\inf_{u\in U} DW_{\theta^*}(x)f(x,u)$ may have incorrect sign in a region around 0

UNIVERSITÄT BAYREUTH

The singularity problem

An approximate (control) Lyapunov function W_{θ^*} will only satisfy the inequalities

$$-\varepsilon + \alpha_1(\|x\|) \leq W_{\theta^*}(x) \leq \alpha_2(\|x\|) + \varepsilon$$

$$\inf_{u \in U} DW_{\theta^*}(x) f(x,u) \leq -\alpha_3(\|x\|) + \varepsilon$$

Since $\alpha_i(\|x\|) \approx 0$ for $x \approx 0$, this means that W_{θ^*} and $\inf_{u \in U} DW_{\theta^*}(x) f(x,u)$ may have incorrect sign in a region around 0

$\rightsquigarrow$ W_{θ^*} is only a Lyapunov function outside this region

The singularity problem

With neural networks as approximation architecture, this problem can be avoided:

The singularity problem

With neural networks as approximation architecture, this problem can be avoided:

For many systems, it is known that V is locally quadratic around 0

The singularity problem

With neural networks as approximation architecture, this problem can be avoided:

For many systems, it is known that V is locally quadratic around 0

Idea [Barreau/Bastianello '25]: Represent the Lyapunov function as

$$V(x) \approx x^T P x + \sum_{i \in \mathcal{I}} W_{\theta,i}(x) \prod_{k=1}^{n} x_k^{i_k}$$

where $W_{\theta,i}(x)$, $i \in \mathcal{I}$ are the (scalar) output components of a neural network with high-dimensional output and

$$\mathcal{I} = \{i = (i_1, i_2, \dots, i_n) \in \{0,1,2,3\}^n \,|\, \sum_{k=1}^{n} i_k = 3\}$$

UNIVERSITÄT BAYREUTH

The singularity problem

With neural networks as approximation architecture, this problem can be avoided:

For many systems, it is known that V is locally quadratic around 0

Idea [Barreau/Bastianello '25]: Represent the Lyapunov function as

$$V(x) \approx x^T P x + \sum_{i \in \mathcal{I}} W_{\theta,i}(x) \prod_{k=1}^{n} x_k^{i_k}$$

where $W_{\theta,i}(x)$, $i \in \mathcal{I}$ are the (scalar) output components of a neural network with high-dimensional output and

$$\mathcal{I} = \{ i = (i_1, i_2, \ldots, i_n) \in \{0,1,2,3\}^n \mid \sum_{k=1}^{n} i_k = 3 \}$$

Alternative approach: use more sampling points around 0 and include 0 in the sampling points

Verification

Verification

A challenging problem is to verify whether a function represented by a neural network is indeed a Lyapunov function

Verification

A challenging problem is to verify whether a function represented by a neural network is indeed a Lyapunov function, i.e., that it satisfies the desired inequalities *for all* x from a given set

Verification

A challenging problem is to verify whether a function represented by a neural network is indeed a Lyapunov function, i.e., that it satisfies the desired inequalities *for all* x from a given set

While there are several papers in this direction, e.g. [Hafstein et al. 14ff; Liu et al. 24ff], using techniques from approximation theory or from formal verification, all of them seem to be prone to the curse of dimensionality

Nonsmooth clfs

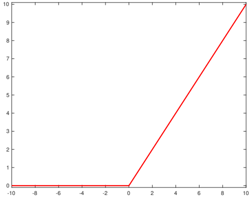
Nonsmooth clfs

As already mentioned, some systems do not allow for a smooth clf

Nonsmooth clfs

As already mentioned, some systems do not allow for a smooth clf

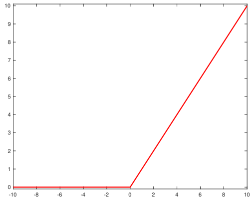
Yet, one of the most popular classes of activation functions for deep neural networks these days are ReLU functions



Nonsmooth clfs

As already mentioned, some systems do not allow for a smooth clf

Yet, one of the most popular classes of activation functions for deep neural networks these days are ReLU functions

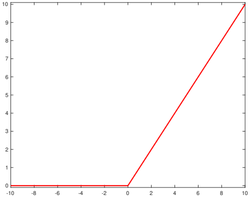


ReLU networks represent nonsmooth functions, more precisely continuous and piecewise affine functions

Nonsmooth clfs

As already mentioned, some systems do not allow for a smooth clf

Yet, one of the most popular classes of activation functions for deep neural networks these days are ReLU functions



ReLU networks represent nonsmooth functions, more precisely continuous and piecewise affine functions

Can we exploit this property?

Nonsmooth clfs

Theorem: Assume there exists a clf that is the minimum of finitely many C^2 functions

Nonsmooth clfs

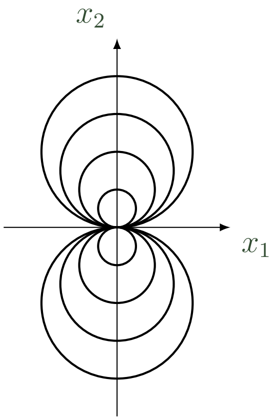
Theorem: Assume there exists a clf that is the minimum of finitely many C^2 functions. Then there exists a clf that can be represented outside an ε-neighbourhood of 0 by a ReLU neural network with at most $\lceil \log_2(n+1) \rceil + 1$ layers

Nonsmooth clfs

Theorem: Assume there exists a clf that is the minimum of finitely many C^2 functions. Then there exists a clf that can be represented outside an ε-neighbourhood of 0 by a ReLU neural network with at most $\lceil \log_2(n+1) \rceil + 1$ layers

Example:

$$\dot{x} = f(x,u) = \begin{pmatrix} (-x_1^2 + x_2^2)u \\ -2x_1x_2u \end{pmatrix}$$

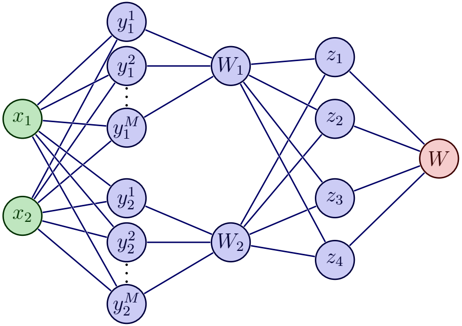


Nonsmooth clfs

Theorem: Assume there exists a clf that is the minimum of finitely many C^2 functions. Then there exists a clf that can be represented outside an ε-neighbourhood of 0 by a ReLU neural network with at most $\lceil \log_2(n+1) \rceil + 1$ layers

Example:

$$\dot{x} = f(x,u) = \begin{pmatrix} (-x_1^2 + x_2^2)u \\ -2x_1x_2u \end{pmatrix}$$

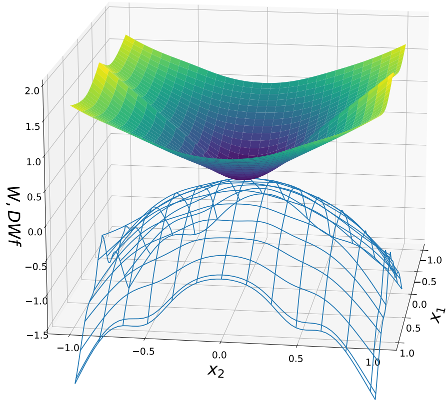


Nonsmooth clfs

Theorem: Assume there exists a clf that is the minimum of finitely many C^2 functions. Then there exists a clf that can be represented outside an ε-neighbourhood of 0 by a ReLU neural network with at most $\lceil \log_2(n+1) \rceil + 1$ layers

Example:

$$\dot{x} = f(x,u) = \begin{pmatrix} (-x_1^2 + x_2^2)u \\ -2x_1x_2u \end{pmatrix}$$

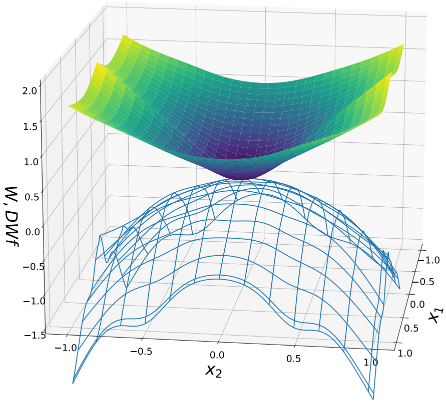


Nonsmooth clfs

Theorem: Assume there exists a clf that is the minimum of finitely many C^2 functions. Then there exists a clf that can be represented outside an ε-neighbourhood of 0 by a ReLU neural network with at most $\lceil \log_2(n+1) \rceil + 1$ layers

Example:

$$\dot{x} = f(x,u) = \begin{pmatrix} (-x_1^2 + x_2^2)u \\ -2x_1x_2u \end{pmatrix}$$



But: learning nonsmooth functions is a challenge

Challenges

- fast and reliable learning algorithms
  - for nonsmooth functions
  - but also in general

Challenges

- fast and reliable learning algorithms
  - for nonsmooth functions
  - but also in general
- computation of the minimum over u
  - ideas from reinforcement learning may help:
  - actor-critic network, policy gradient method

Challenges

- fast and reliable learning algorithms
  - for nonsmooth functions
  - but also in general
- computation of the minimum over u
  - ideas from reinforcement learning may help:
  - actor-critic network, policy gradient method
- efficient verification of a clf candidate
  - if possible avoiding the curse of dimensionality

Challenges

- fast and reliable learning algorithms
  - for nonsmooth functions
  - but also in general
- computation of the minimum over u
  - ideas from reinforcement learning may help:
  - actor-critic network, policy gradient method
- efficient verification of a clf candidate
  - if possible avoiding the curse of dimensionality
- extension to, e.g., control barrier functions
  - preliminary work of Jun Liu et al. and others exists

References

Mario Sperl, Jonas Mysliwitz, Lars Grüne, *On the existence and neural network representation of separable control Lyapunov functions*, Automatica, 182 (2025), 112517

Kaiwen Chen and Alessandro Astolfi, *Active Nodes of Network Systems With Sum-Type Dissipation Inequalities*, IEEE Transactions on Automatic Control 69 (2024), 3896–3911

Matthieu Barreau, Nicola Bastianello, *Learning and verifying maximal Taylor-neural Lyapunov functions*, Preprint on ResearchGate

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, Ruikun Zhou, *Physics-informed neural network Lyapunov functions: PDE characterization, learning, and verification*, Automatica, 175 (2025), 112193

Lars Grüne, Mario Sperl, Debasish Chatterjee, *Representation of practical nonsmooth control Lyapunov functions by piecewise affine functions and neural networks*, Systems & Control Letters, 202 (2025), 106103