Decaying sensitivity and separable optimal value functions

Lars Grüne

Mathematical Institute, University of Bayreuth, Germany

based on joint work with
Dante Kalise (London), Luca Saluzzi (Pisa), Mario Sperl (Bayreuth)

supported by DFG Deutsche Forschungsgemeinschaft

Nonlinear partial differential equations: theory, numerics and applications
A conference in memory of Maurizio Falcone
Rome, 24–26 May 2023

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

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

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

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

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

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

Objective:

$$\underset{u \in \mathcal{U}}{\text{minimize}} \; J(x_0, u) := \int_0^\infty \ell(x(t), u(t))\, dt \quad \text{or} \quad J(x_0, u) := \sum_{t=0}^{\infty} \ell(x(t), u(t))$$

Objective

$$\underset{u \in \mathcal{U}}{\text{minimize}} \; J(x_0, u) := \int_0^\infty \ell(x(t), u(t))\,dt \quad \text{or} \quad J(x_0, u) := \sum_{t=0}^{\infty} \ell(x(t), u(t))$$

Objective

$$\underset{u\in\mathcal{U}}{\text{minimize}}\ J(x_0,u) := \int_0^\infty \ell(x(t),u(t))\,dt \quad \text{or} \quad J(x_0,u) := \sum_{t=0}^{\infty} \ell(x(t),u(t))$$

For this problem, an (approximately) optimal feedback control can be computed from (an approximation of) the optimal value function

$$V(x_0) := \inf_{u\in\mathcal{U}} J(x_0,u)$$

via the associated Hamilton-Jacobi-Bellman equation

$$\sup_{u\in U}\{-DV(x)f(x,u) - \ell(x,u)\} = 0$$

or the Bellman equation

$$\sup_{u\in U}\{V(x) - V(f(x,u)) - \ell(x,u)\} = 0$$

Curse of Dimensionality

Curse of Dimensionality

- Challenge: Common numerical methods suffer from the curse of dimensionality, i.e., an exponential growth of the computational effort in the state dimension

Curse of Dimensionality

- Challenge: Common numerical methods suffer from the curse of dimensionality, i.e., an exponential growth of the computational effort in the state dimension
- Known fact: Deep Neural Networks (DNNs) are capable of overcoming the curse of dimensionality for functions with certain beneficial structures

Curse of Dimensionality

- Challenge: Common numerical methods suffer from the curse of dimensionality, i.e., an exponential growth of the computational effort in the state dimension
- Known fact: Deep Neural Networks (DNNs) are capable of overcoming the curse of dimensionality for functions with certain beneficial structures
- Goal: Detect and exploit such structures for approximating control Lyapunov functions and optimal value functions

Simplified setting

Instead of looking for solutions of the equations

$$\sup_{u\in U}\{-DV(x)f(x,u)-\ell(x,u)\} = 0 \quad \text{or} \quad \sup_{u\in U}\{V(x)-V(f(x,u))-\ell(x,u)\} = 0$$

we start by computing supersolutions

$$\sup_{u\in U}\{-DV(x)f(x,u)-\ell(x,u)\} \geq 0 \quad \text{or} \quad \sup_{u\in U}\{V(x)-V(f(x,u))-\ell(x,u)\} \geq 0$$

UNIVERSITÄT BAYREUTH

Simplified setting

Instead of looking for solutions of the equations

$$\sup_{u\in U}\{-DV(x)f(x,u)-\ell(x,u)\} = 0 \quad \text{or} \quad \sup_{u\in U}\{V(x)-V(f(x,u))-\ell(x,u)\} = 0$$

we start by computing supersolutions

$$\sup_{u\in U}\{-DV(x)f(x,u)-\ell(x,u)\} \geq 0 \quad \text{or} \quad \sup_{u\in U}\{V(x)-V(f(x,u))-\ell(x,u)\} \geq 0$$

These are interesting in their own right, because they describe control Lyapunov functions and, in addition, upper bounds for the value functions

Lyapunov functions

Lyapunov functions

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 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 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

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

Lyapunov functions

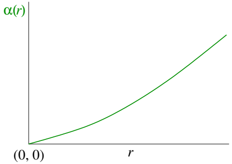
We consider autonomous ordinary differential equations (ODEs)

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

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 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

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

Lyapunov functions

We consider autonomous ordinary differential equations (ODEs)

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

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

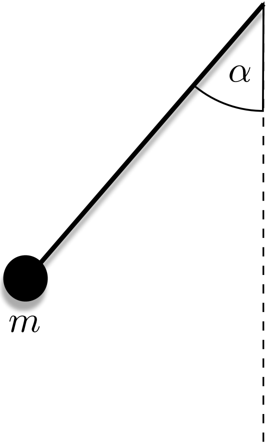
Assume $x^* = 0$ is an equilibrium, i.e., $f(0) = 0$

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

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

UNIVERSITÄT BAYREUTH

Example: Mathematical Pendulum



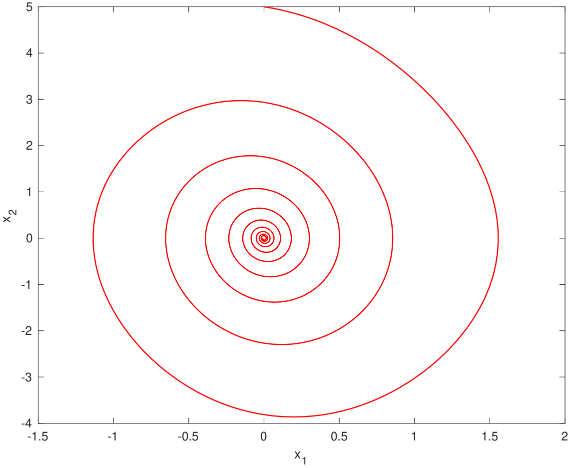
$x_1 = \alpha =$ angle

$x_2 =$ angular velocity

$\rightsquigarrow$ ordinary differential equation

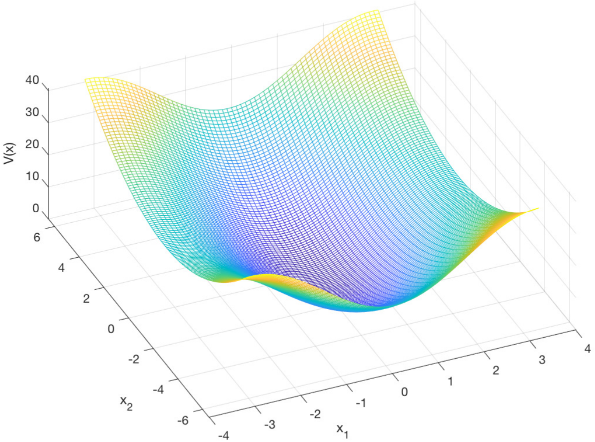
$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -g\sin(x_1) - \frac{k}{m}x_2 \end{aligned}$$

Pendulum solution and Lyapunov function



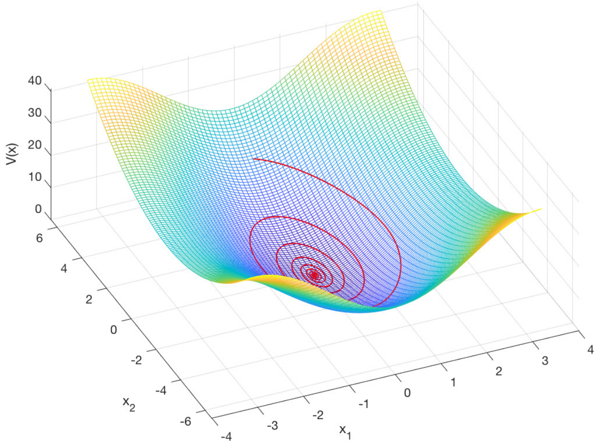
Solution of pendulum equation

Pendulum solution and Lyapunov function



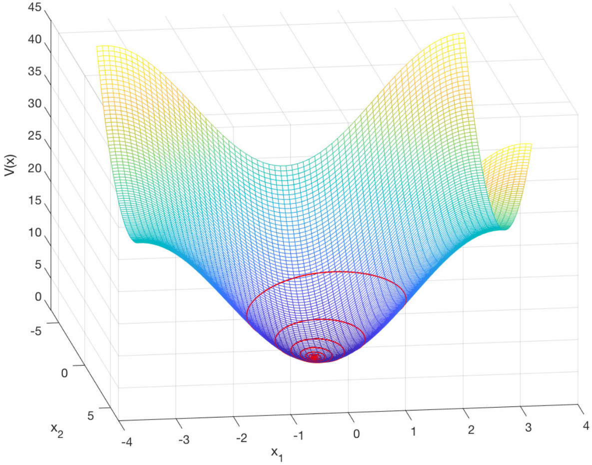
Lyapunov function $V(x) = x_2^2/2 + g(1 - \cos x_1) + 0.1 x_2 \sin(x_1)$

Pendulum solution and Lyapunov function



Lyapunov function with solution superimposed

Pendulum solution and Lyapunov function



Lyapunov function with solution superimposed

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

Numerical computation of Lyapunov functions

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

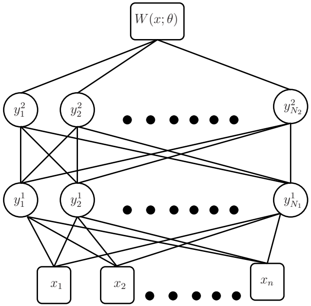
- Series expansion [Kirin et al. '82]
- Semi-Lagrangian schemes [Camilli/Gr./Wirth '00, Falcone/Gr./Wirth '00]
- Finite elements and linear programming [Hafstein '02ff]
- Sum-of-squares methods [Papachristodoulou/Prajna '02]
- Radial Basis functions [Giesl '04ff, Giesl/Wendland '07ff]

All these methods suffer from the curse of dimensionality

Can deep neural networks do better?

Deep neural networks

Deep neural network with 2 hidden layers



output $W(x;\theta) = a \cdot y_k^2 + c$

$\ell = 2$ $y_k^2 = \sigma^2(w_k^2 \cdot y^1 + b_k^2)$

$\ell = 1$ $y_k^1 = \sigma^1(w_k^1 \cdot x + b_k^1)$

input

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

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

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

θ = vector of all parameters $(w_k^\ell, b_k^\ell, a, c)$

$W(x;\theta^*) \approx V(x)$ approx. Lyapunov function for "trained" θ^*

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

  $\rightsquigarrow$ Unlikely to work for deterministic problems

Beneficial structures

- Compositional functions, cf. e.g. [Poggio/Mhaskar/Rosasco/Miranda/Liao '17, Kang/Gong '22, Dahmen '23]

Beneficial structures

- Compositional functions, cf. e.g. [Poggio/Mhaskar/Rosasco/Miranda/Liao '17, Kang/Gong '22, Dahmen '23]. These are functions of the form

$$g(x) = g_1(g_2(x_{i_1}, x_{i_2}), g_3(x_{i_3}) + g_4(g_5(g_6(x_{i_4}, x_{i_5}))) + \dots$$

where each component function g_i depends only on a number of arguments m that is independent of n

Beneficial structures

- Compositional functions, cf. e.g. [Poggio/Mhaskar/Rosasco/Miranda/Liao '17, Kang/Gong '22, Dahmen '23]. These are functions of the form

$$g(x) = g_1(g_2(x_{i_1}, x_{i_2}), g_3(x_{i_3}) + g_4(g_5(g_6(x_{i_4}, x_{i_5}))) + \dots$$

  where each component function g_i depends only on a number of arguments m that is independent of n

A particular example are separable functions

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

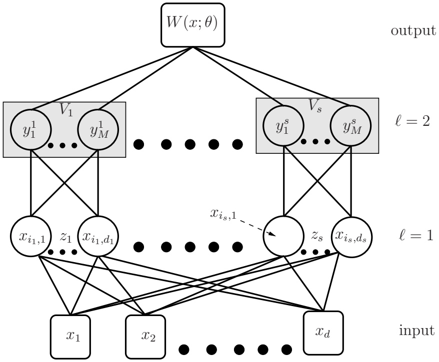
with m bounded independent of n and $s \leq n$

Why are separable functions beneficial?

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

Why are separable functions beneficial?

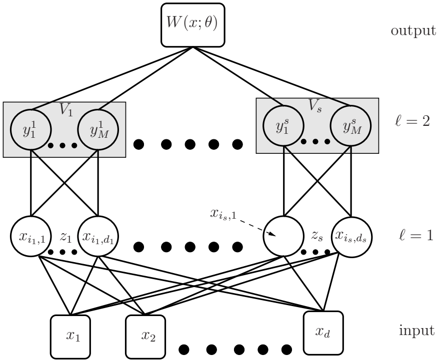
Separable function: $V(x) = \sum_{j=1}^{s} V_j(z_j), \quad z_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$



Why are separable functions beneficial?

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

We can approximate the individual V_j by the grey blocks

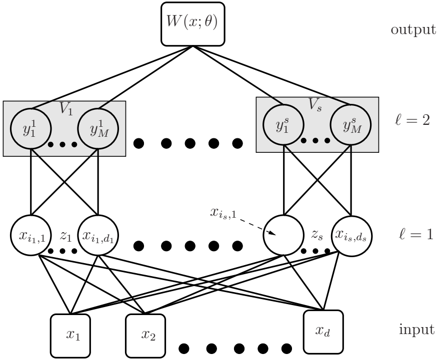


Why are separable functions beneficial?

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

We can approximate the individual V_j by the grey blocks

When the dimension n grows, the number of blocks grows linearly



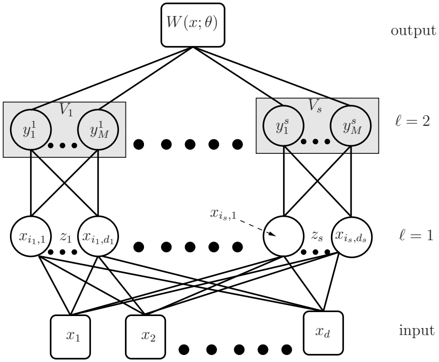
Why are separable functions beneficial?

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

We can approximate the individual V_j by the grey blocks

When the dimension n grows, the number of blocks grows linearly

If the d_j's are constant or grow only slowly with n, the number of neurons in each block also grows slowly



Why are separable functions beneficial?

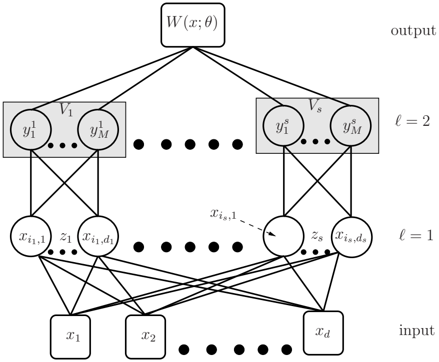
Separable function: $V(x) = \sum_{j=1}^{s} V_j(z_j), \quad z_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$

We can approximate the individual V_j by the grey blocks

When the dimension n grows, the number of blocks grows linearly

If the d_j's are constant or grow only slowly with n, the number of neurons in each block also grows slowly

$\rightsquigarrow$ no curse of dimensionality



Why are separable functions beneficial?

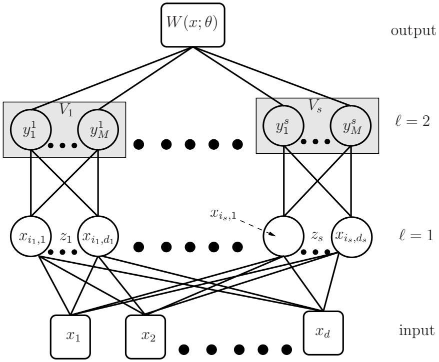
Separable function: $V(x) = \sum_{j=1}^{s} V_j(z_j), \quad z_j = \begin{pmatrix} x_{i_{j,1}} \\ \vdots \\ x_{i_{j,d_j}} \end{pmatrix}$

We can approximate the individual V_j by the grey blocks

When the dimension n grows, the number of blocks grows linearly

If the d_j's are constant or grow only slowly with n, the number of neurons in each block also grows slowly

$\rightsquigarrow$ no curse of dimensionality



In the first layer we can even implement more complex transformations than merely splitting up x into the z_j

Nonlinear small-gain theory

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

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

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

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

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

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}

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

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

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

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

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, ...]

Complexity theorem

Theorem [Gr. 21]: Lyapunov functions $V(x) = \sum_{j=1}^{s} V_j(T_j x)$ with $d_j = \text{rank} T_j \leq d_{\max}$ independent of n can be approximated with any accuracy $\varepsilon > 0$ with a number of neurons growing only polynomially in n

Complexity theorem

Theorem [Gr. 21]: Lyapunov functions $V(x) = \sum_{j=1}^{s} V_j(T_j x)$ with $d_j = \text{rank} T_j \leq d_{\max}$ independent of n can be approximated with any accuracy $\varepsilon > 0$ with a number of neurons growing only polynomially in n

Note: The small-gain theory guarantees the existence of a compositional Lyapunov functions, but for using this result with DNNs, neither the z_j nor the V_j need to be known in advance

Complexity theorem

Theorem [Gr. 21]: Lyapunov functions $V(x) = \sum_{j=1}^{s} V_j(T_j x)$ with $d_j = \text{rank} T_j \leq d_{\max}$ independent of n can be approximated with any accuracy $\varepsilon > 0$ with a number of neurons growing only polynomially in n

Note: The small-gain theory guarantees the existence of a compositional Lyapunov functions, but for using this result with DNNs, neither the z_j nor the V_j need to be known in advance

Using an appropriate training algorithm, the network will "learn" this structure during the training process

10d Numerical Example

$$\dot{x}(t) = T^{-1}\hat{f}(x)(Tx)$$

with

$$\hat{f}(x) = \begin{pmatrix} -x_1 + 0.5x_2 - 0.1x_9^2 \\ -0.5x_1 - x_2 \\ -x_3 + 0.5x_4 - 0.1x_1^2 \\ -0.5x_3 - x_4 \\ -x_5 + 0.5x_6 + 0.1x_7^2 \\ -0.5x_5 - x_6 \\ -x_7 + 0.5x_8 \\ -0.5x_7 - x_8 \\ -x_9 + 0.5x_{10} \\ -0.5x_9 - x_{10} + 0.1x_2^2 \end{pmatrix}, \qquad T = \begin{pmatrix} -\frac{1}{5} & -\frac{3}{10} & \frac{1}{2} & -\frac{4}{5} & \frac{4}{5} & \frac{2}{5} & \frac{7}{10} & \frac{7}{10} & -1 & \frac{4}{5} \\ \frac{1}{5} & 1 & \frac{9}{10} & \frac{4}{5} & -\frac{1}{10} & \frac{3}{5} & -\frac{3}{10} & \frac{1}{2} & \frac{4}{5} & -\frac{3}{10} \\ -\frac{3}{10} & \frac{3}{10} & \frac{2}{5} & -\frac{2}{5} & 0 & -\frac{3}{5} & \frac{3}{10} & \frac{3}{5} & 1 & -\frac{1}{2} \\ -\frac{7}{10} & -\frac{1}{10} & -\frac{3}{5} & -\frac{1}{5} & -\frac{3}{5} & \frac{2}{5} & \frac{1}{10} & -\frac{1}{10} & \frac{1}{10} & -\frac{3}{5} \\ \frac{1}{10} & -\frac{3}{5} & -\frac{9}{10} & -\frac{7}{10} & -\frac{1}{5} & -\frac{1}{10} & \frac{1}{10} & \frac{1}{5} & 0 & -\frac{4}{5} \\ \frac{3}{5} & \frac{9}{10} & -\frac{1}{5} & 1 & \frac{2}{5} & \frac{1}{2} & 0 & -\frac{1}{10} & -\frac{2}{5} & 0 \\ -1 & 1 & \frac{7}{10} & \frac{3}{5} & -\frac{4}{5} & -\frac{4}{5} & 0 & -\frac{1}{5} & -\frac{1}{5} & \frac{7}{10} \\ -\frac{9}{10} & \frac{4}{5} & \frac{1}{5} & 1 & -\frac{4}{5} & \frac{2}{5} & -\frac{3}{10} & \frac{7}{10} & \frac{1}{5} & -\frac{4}{5} \\ \frac{3}{5} & -\frac{1}{10} & -\frac{2}{5} & -\frac{1}{2} & -\frac{3}{10} & -\frac{1}{10} & -\frac{7}{10} & 1 & \frac{4}{5} & -\frac{3}{10} \\ 0 & -1 & -\frac{1}{10} & \frac{2}{5} & -\frac{3}{10} & -\frac{1}{10} & -\frac{1}{5} & \frac{7}{10} & -\frac{1}{10} & \frac{4}{5} \end{pmatrix}$$

10d Numerical Example

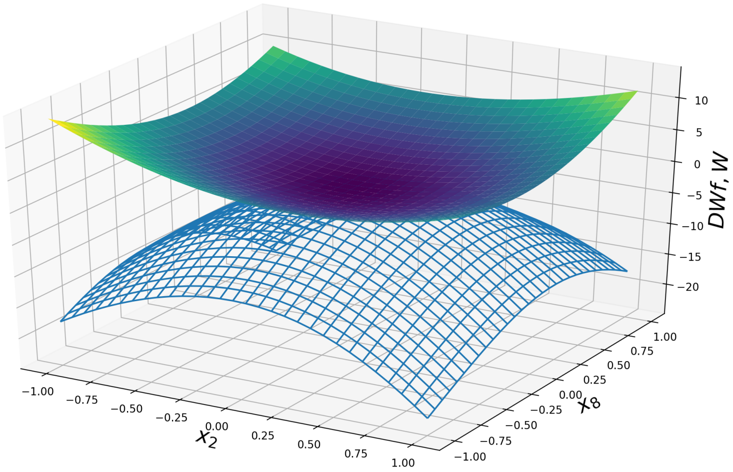
$$\dot{x}(t) = T^{-1}\hat{f}(x)(Tx)$$

with

$$\hat{f}(x) = \begin{pmatrix} -x_1 + 0.5x_2 - 0.1x_9^2 \\ -0.5x_1 - x_2 \\ -x_3 + 0.5x_4 - 0.1x_1^2 \\ -0.5x_3 - x_4 \\ -x_5 + 0.5x_6 + 0.1x_7^2 \\ -0.5x_5 - x_6 \\ -x_7 + 0.5x_8 \\ -0.5x_7 - x_8 \\ -x_9 + 0.5x_{10} \\ -0.5x_9 - x_{10} + 0.1x_2^2 \end{pmatrix}, \qquad T = \begin{pmatrix} -\frac{1}{5} & -\frac{3}{10} & \frac{1}{2} & -\frac{4}{5} & \frac{4}{5} & \frac{2}{5} & \frac{7}{10} & \frac{7}{10} & -1 & \frac{4}{5} \\ \frac{1}{5} & 1 & \frac{9}{10} & \frac{4}{5} & -\frac{1}{10} & \frac{3}{5} & -\frac{3}{10} & \frac{1}{2} & \frac{4}{5} & -\frac{3}{10} \\ -\frac{3}{10} & \frac{3}{10} & \frac{2}{5} & -\frac{2}{5} & 0 & -\frac{3}{5} & \frac{3}{10} & \frac{3}{5} & 1 & -\frac{1}{2} \\ -\frac{7}{10} & -\frac{1}{10} & -\frac{3}{5} & -\frac{1}{5} & -\frac{3}{5} & \frac{2}{5} & \frac{1}{10} & -\frac{1}{10} & \frac{1}{10} & -\frac{3}{5} \\ \frac{1}{10} & -\frac{3}{5} & -\frac{9}{10} & -\frac{7}{10} & -\frac{1}{5} & -\frac{1}{10} & \frac{1}{10} & \frac{1}{5} & 0 & -\frac{4}{5} \\ \frac{3}{5} & \frac{9}{10} & -\frac{1}{5} & 1 & \frac{2}{5} & \frac{1}{2} & 0 & -\frac{1}{10} & -\frac{2}{5} & 0 \\ -1 & 1 & \frac{7}{10} & \frac{3}{5} & -\frac{4}{5} & -\frac{4}{5} & 0 & -\frac{1}{5} & -\frac{1}{5} & \frac{7}{10} \\ -\frac{9}{10} & \frac{4}{5} & \frac{1}{5} & 1 & -\frac{4}{5} & \frac{2}{5} & -\frac{3}{10} & \frac{7}{10} & \frac{1}{5} & -\frac{4}{5} \\ \frac{3}{5} & -\frac{1}{10} & -\frac{2}{5} & -\frac{1}{2} & -\frac{3}{10} & -\frac{1}{10} & -\frac{7}{10} & 1 & \frac{4}{5} & -\frac{3}{10} \\ 0 & -1 & -\frac{1}{10} & \frac{2}{5} & -\frac{3}{10} & -\frac{1}{10} & -\frac{1}{5} & \frac{7}{10} & -\frac{1}{10} & \frac{4}{5} \end{pmatrix}$$

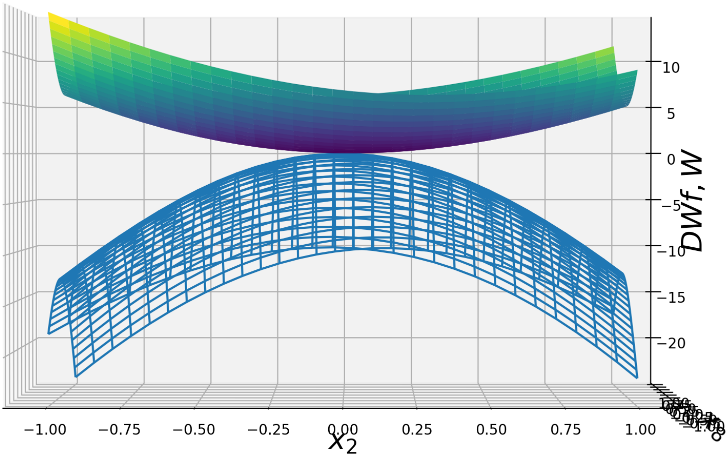
We perform the training with a network with 5 sublayers with dimension $d_{max} = 2$ ($\rightsquigarrow$ 2 671 parameters), and $m = 400\,000$ test points

10d Example



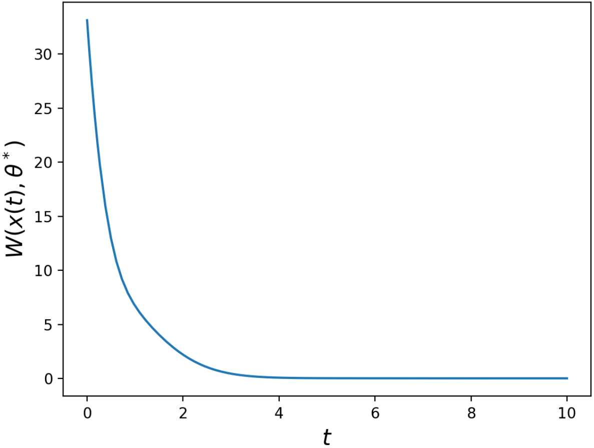
10
5
0
−5
−10
−15
−20
DWf, W
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x_2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
x_8

10d Example



Computation time: 266s

10d Example – Evaluation along trajectories



Initial value $x_0 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)^T$

Control Lyapunov functions

Control Lyapunov functions

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

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

UNIVERSITÄT BAYREUTH

Control Lyapunov functions

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

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

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

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

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

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

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

Question: When can we employ small gain techniques here?

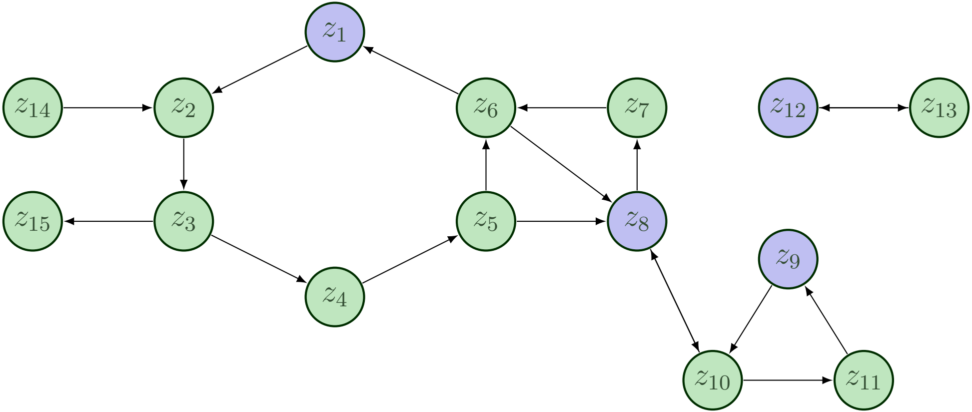
Recall the sufficient condition

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

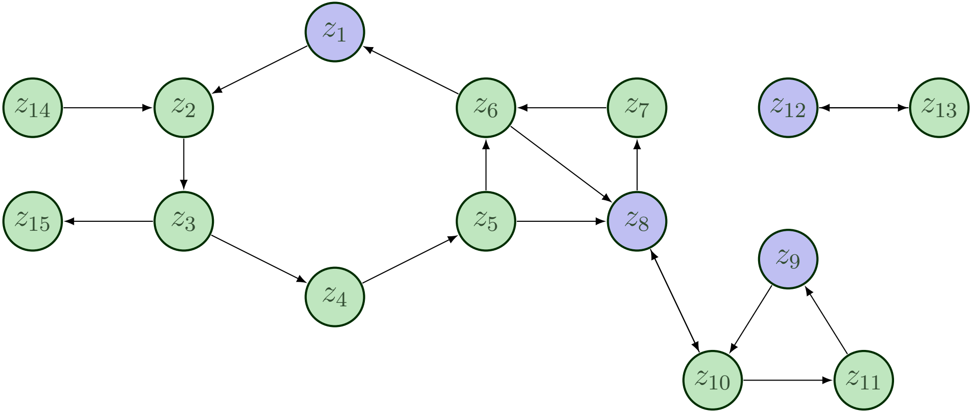
This implies:

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 '20]

Example for a suitable graph structure



Example for a suitable graph structure

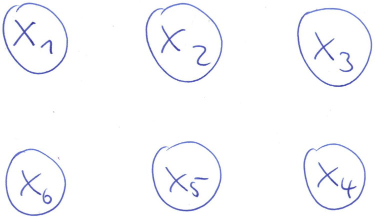


$$\rightsquigarrow \quad V(x) = \sum_{j=1}^{15} V_j(z_j)$$

Computation with DNN

Example:

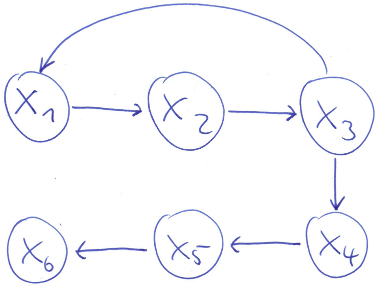
$$\begin{aligned}\dot{x}_1 &= x_3 + u\\ \dot{x}_2 &= x_1 - x_2 + x_1^2\\ \dot{x}_3 &= x_2 - x_3\\ \dot{x}_4 &= x_3 - x_4\\ \dot{x}_5 &= x_4 - x_5\\ \dot{x}_6 &= x_5 - x_6\end{aligned}$$



Computation with DNN

Example:

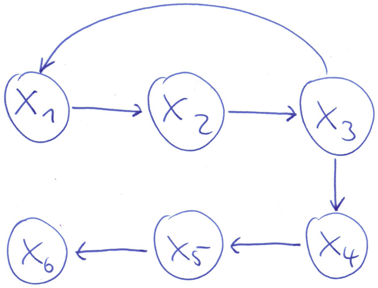
$$
\begin{aligned}
\dot{x}_1 &= x_3 + u \\
\dot{x}_2 &= x_1 - x_2 + x_1^2 \\
\dot{x}_3 &= x_2 - x_3 \\
\dot{x}_4 &= x_3 - x_4 \\
\dot{x}_5 &= x_4 - x_5 \\
\dot{x}_6 &= x_5 - x_6
\end{aligned}
$$



Computation with DNN

Example:

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



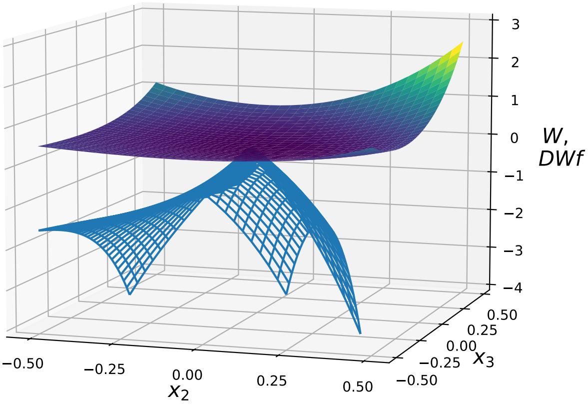
$\rightsquigarrow$ $$V(x) = \sum_{j=1}^{6} V(x_j)$$

Computation with DNN

Example:

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

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



Computation time: 820s

(Approximately) Optimal Value Functions

Optimal Value Functions

In optimal control, we want to solve

$$\sup_{u \in U} \{-DV(x)f(x,u) - \ell(x,u)\} = 0$$

in continuous time or

$$\sup_{u \in U} \{V(x) - V(f(x,u)) - \ell(x,u)\} = 0$$

in discrete time

UNIVERSITÄT BAYREUTH

Optimal Value Functions

In optimal control, we want to solve

$$\sup_{u \in U}\{-DV(x)f(x,u) - \ell(x,u)\} = 0$$

in continuous time or

$$\sup_{u \in U}\{V(x) - V(f(x,u)) - \ell(x,u)\} = 0$$

in discrete time

Assuming that an optimal value function V or an approximation thereof has a separable structure may be too demanding, even after coordinate transformations

Seperable supersolutions are also likely to be very suboptimal

Optimal Value Functions

In optimal control, we want to solve

$$\sup_{u\in U}\{-DV(x)f(x,u)-\ell(x,u)\}=0$$

in continuous time or

$$\sup_{u\in U}\{V(x)-V(f(x,u))-\ell(x,u)\}=0$$

in discrete time

Assuming that an optimal value function V or an approximation thereof has a separable structure may be too demanding, even after coordinate transformations

Seperable supersolutions are also likely to be very suboptimal

This is because optimisation usually exploits the interaction between subsystems

Optimal Value Functions

In optimal control, we want to solve

$$\sup_{u\in U}\{-DV(x)f(x,u)-\ell(x,u)\}=0$$

in continuous time or

$$\sup_{u\in U}\{V(x)-V(f(x,u))-\ell(x,u)\}=0$$

in discrete time

Assuming that an optimal value function V or an approximation thereof has a separable structure may be too demanding, even after coordinate transformations

Seperable supersolutions are also likely to be very suboptimal

This is because optimisation usually exploits the interaction between subsystems

Remedy: Overlapping decompositions offer more flexibility

Decaying sensitivity

Consider, as before, a decomposition of x into subvectors z_j corresponding to subsystems connected via a graph

UNIVERSITÄT BAYREUTH

Decaying sensitivity

Consider, as before, a decomposition of x into subvectors z_j corresponding to subsystems connected via a graph. Assume the Lipschitz constant of the map

$$z_k \mapsto V(z_1, \ldots, z_{l-1}, z_l, z_{l+1}, \ldots, z_s) - V(z_1, \ldots, z_{l-1}, 0, z_{l+1}, \ldots, z_s) \quad (*)$$

decreases to 0 with the distance of the k-th and the l-th subsystem in the graph

Decaying sensitivity

Consider, as before, a decomposition of x into subvectors z_j corresponding to subsystems connected via a graph. Assume the Lipschitz constant of the map

$$z_k \mapsto V(z_1, \ldots, z_{l-1}, z_l, z_{l+1}, \ldots, z_s) - V(z_1, \ldots, z_{l-1}, 0, z_{l+1}, \ldots, z_s) \quad (*)$$

decreases to 0 with the distance of the k-th and the l-th subsystem in the graph

Then V can be approximated by a sum of functions, each with a bounded number of arguments z_j [Sperl/Saluzzi/Gr./Kalise '23]

UNIVERSITÄT BAYREUTH

Decaying sensitivity

Consider, as before, a decomposition of x into subvectors z_j corresponding to subsystems connected via a graph. Assume the Lipschitz constant of the map

$$z_k \mapsto V(z_1,\ldots,z_{l-1},z_l,z_{l+1},\ldots,z_s) - V(z_1,\ldots,z_{l-1},0,z_{l+1},\ldots,z_s) \qquad (*)$$

decreases to 0 with the distance of the k-th and the l-th subsystem in the graph

Then V can be approximated by a sum of functions, each with a bounded number of arguments z_j (details in a minute) [Sperl/Saluzzi/Gr./Kalise '23]

Decaying sensitivity

Consider, as before, a decomposition of x into subvectors z_j corresponding to subsystems connected via a graph. Assume the Lipschitz constant of the map

$$z_k \mapsto V(z_1, \ldots, z_{l-1}, z_l, z_{l+1}, \ldots, z_s) - V(z_1, \ldots, z_{l-1}, 0, z_{l+1}, \ldots, z_s) \qquad (*)$$

decreases to 0 with the distance of the k-th and the l-th subsystem in the graph

Then V can be approximated by a sum of functions, each with a bounded number of arguments z_j (details in a minute) [Sperl/Saluzzi/Gr./Kalise '23]

Note:
- The same z_j may appear in several of the functions in the sum

Decaying sensitivity

Consider, as before, a decomposition of x into subvectors z_j corresponding to subsystems connected via a graph. Assume the Lipschitz constant of the map

$$z_k \mapsto V(z_1, \ldots, z_{l-1}, z_l, z_{l+1}, \ldots, z_s) - V(z_1, \ldots, z_{l-1}, 0, z_{l+1}, \ldots, z_s) \qquad (*)$$

decreases to 0 with the distance of the k-th and the l-th subsystem in the graph

Then V can be approximated by a sum of functions, each with a bounded number of arguments z_j (details in a minute) [Sperl/Saluzzi/Gr./Kalise '23]

Note:

- The same z_j may appear in several of the functions in the sum
- If V is separable, $(*)$ has Lipschitz constant 0 for all $k \neq l$

UNIVERSITÄT BAYREUTH

Decaying sensitivity

Consider, as before, a decomposition of x into subvectors z_j corresponding to subsystems connected via a graph. Assume the Lipschitz constant of the map

$$z_k \mapsto V(z_1, \ldots, z_{l-1}, z_l, z_{l+1}, \ldots, z_s) - V(z_1, \ldots, z_{l-1}, 0, z_{l+1}, \ldots, z_s) \qquad (*)$$

decreases to 0 with the distance of the k-th and the l-th subsystem in the graph

Then V can be approximated by a sum of functions, each with a bounded number of arguments z_j (details in a minute) [Sperl/Saluzzi/Gr./Kalise '23]

Note:

- The same z_j may appear in several of the functions in the sum
- If V is separable, $(*)$ has Lipschitz constant 0 for all $k \neq l$
- If $V(x) = x^T P x$ is quadratic, the assumption on $(*)$ requires the sub-matrices P_{kl} to decrease with the distance of the k-th and the l-th subsystem in the graph

Decaying sensitivity

Consider, as before, a decomposition of x into subvectors z_j corresponding to subsystems connected via a graph. Assume the Lipschitz constant of the map

$$z_k \mapsto V(z_1, \dots, z_{l-1}, z_l, z_{l+1}, \dots, z_s) - V(z_1, \dots, z_{l-1}, 0, z_{l+1}, \dots, z_s) \qquad (*)$$

decreases to 0 with the distance of the k-th and the l-th subsystem in the graph

Then V can be approximated by a sum of functions, each with a bounded number of arguments z_j (details in a minute) [Sperl/Saluzzi/Gr./Kalise '23]

Note:

- The same z_j may appear in several of the functions in the sum
- If V is separable, $(*)$ has Lipschitz constant 0 for all $k \neq l$
- If $V(x) = x^T P x$ is quadratic, the assumption on $(*)$ requires the sub-matrices P_{kl} to decrease with the distance of the k-th and the l-th subsystem in the graph
- Related to [Shin/Anitescu/Zavala '22, Zhang/Li/Li '22]

Decaying sensitivity

Note: decaying sensitivity is a property of optimal solutions, not of any solution

Decaying sensitivity

Note: decaying sensitivity is a property of optimal solutions, not of any solution

Example: convoy of vehicles

Decaying sensitivity

Note: decaying sensitivity is a property of optimal solutions, not of any solution

Example: convoy of vehicles

It is known that a perturbation in the first vehicle (e.g., a braking manoeuvre) may amplify while propagating through the convoy

Decaying sensitivity

Note: decaying sensitivity is a property of optimal solutions, not of any solution

Example: convoy of vehicles

It is known that a perturbation in the first vehicle (e.g., a braking manoeuvre) may amplify while propagating through the convoy

However, the perturbation will decrease quickly, if the vehicles are controlled optimally

Decaying sensitivity: Example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

UNIVERSITÄT BAYREUTH

Decaying sensitivity: Example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

UNIVERSITÄT BAYREUTH

Decaying sensitivity: Example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

$$\int_0^\infty (x_1(t) - x_{ref}(t))^2 + \sum_{i=1}^{N-1} (x_{i+1}(t) - x_i(t) - L)^2 + \gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2 dt$$

UNIVERSITÄT BAYREUTH

Decaying sensitivity: Example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

$$\int_0^\infty \underbrace{(x_1(t) - x_{ref}(t))^2}_{\text{reference for 1st vehicle}} + \sum_{i=1}^{N-1} (x_{i+1}(t) - x_i(t) - L)^2 + \gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2 dt$$

Decaying sensitivity: Example

Consider a convoy of $i = 1, \dots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

$$\int_0^\infty \underbrace{(x_1(t) - x_{ref}(t))^2}_{\text{reference for 1st vehicle}} + \sum_{i=1}^{N-1} \underbrace{(x_{i+1}(t) - x_i(t) - L)^2}_{\text{desired distance}} + \gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2 dt$$

Decaying sensitivity: Example

Consider a convoy of $i = 1, \ldots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

$$\int_0^\infty \underbrace{(x_1(t) - x_{ref}(t))^2}_{\substack{\text{reference for}\\ \text{1st vehicle}}} + \sum_{i=1}^{N-1} \underbrace{(x_{i+1}(t) - x_i(t) - L)^2}_{\text{desired distance}} + \underbrace{\gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2}_{\text{regularization terms}} \, dt$$

UNIVERSITÄT BAYREUTH

Decaying sensitivity: Example

Consider a convoy of $i = 1, \dots, N$ vehicles on a road with state $z_i = (x_i, v_i)^T$ and dynamics

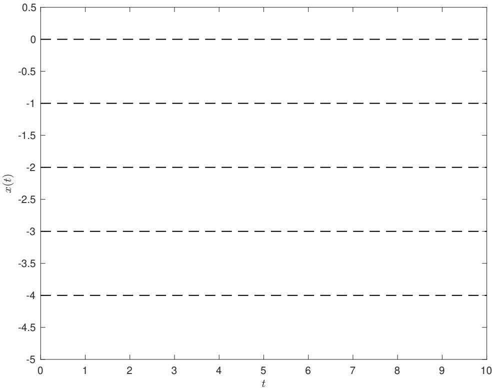
$$\dot{x}_i = v_i, \quad \dot{v}_i = u_i$$

We compute a control that minimizes the functional

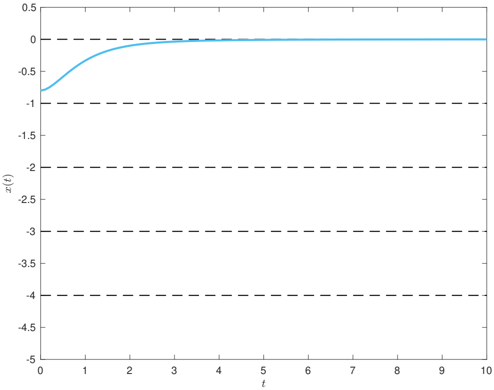
$$\int_0^\infty \underbrace{(x_1(t) - x_{ref}(t))^2}_{\substack{\text{reference for} \\ \text{1st vehicle}}} + \sum_{i=1}^{N-1} \underbrace{(x_{i+1}(t) - x_i(t) - L)^2}_{\text{desired distance}} + \underbrace{\gamma \|v(t) - I v_{ref}\|_2^2 + \delta \|u(t)\|_2^2}_{\text{regularization terms}} \, dt$$

In the simulation: $N = 100$, shown $i = 1, \dots, 5$, $x_{ref} \equiv 0$, $v_{ref} \equiv 0$

Decaying sensitivity: Example

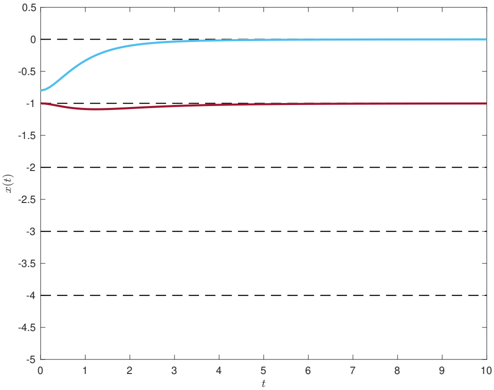


Decaying sensitivity: Example

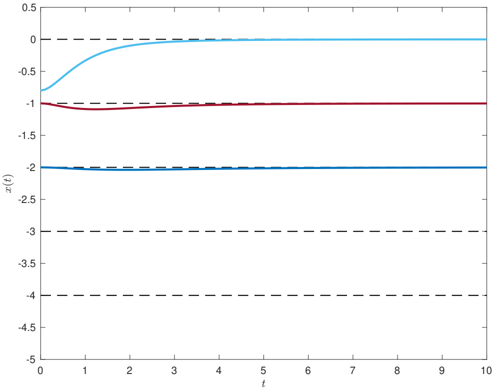


UNIVERSITÄT BAYREUTH

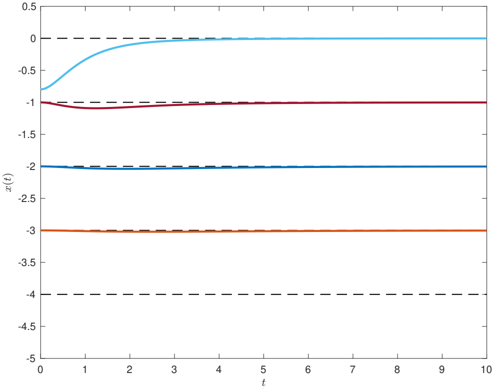
Decaying sensitivity: Example



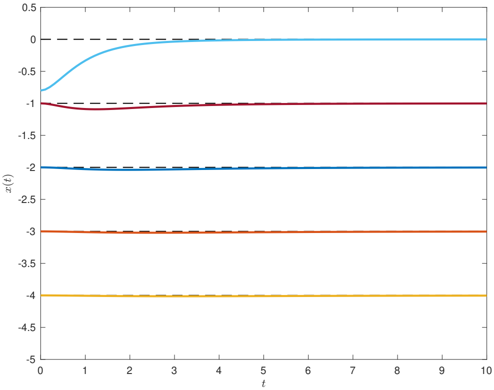
Decaying sensitivity: Example



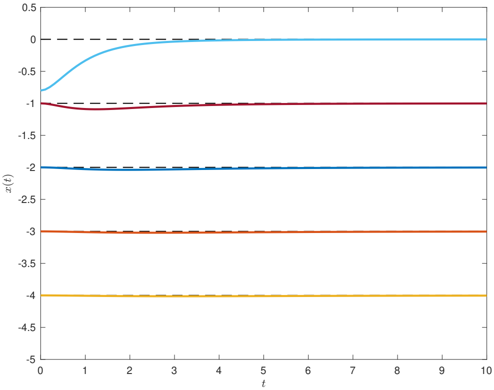
Decaying sensitivity: Example

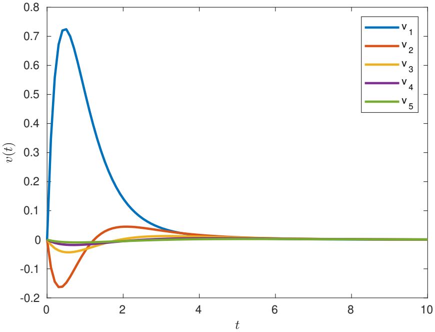


Decaying sensitivity: Example



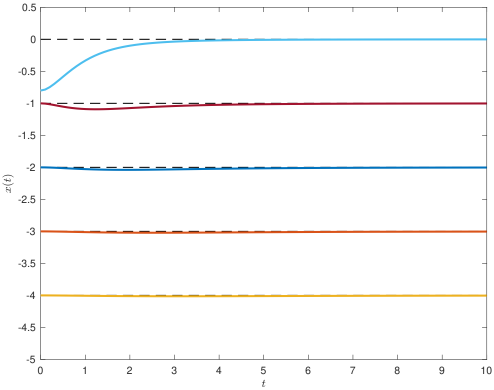
Decaying sensitivity: Example

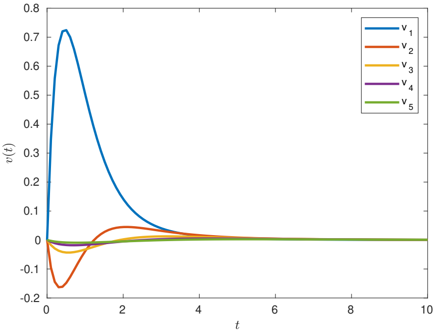




UNIVERSITÄT BAYREUTH

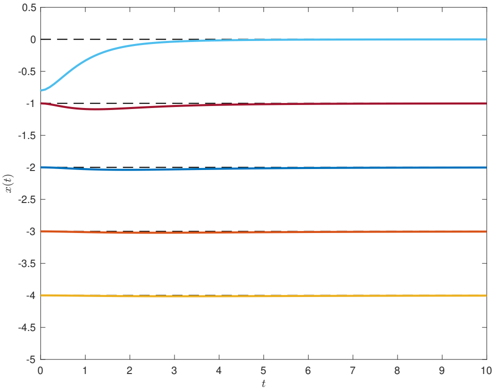
Decaying sensitivity: Example

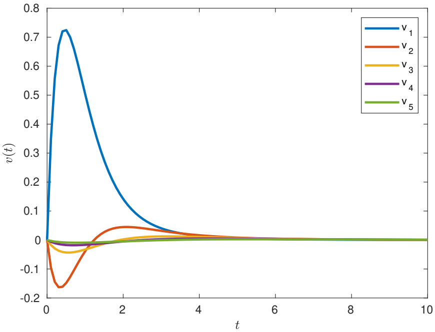




For discrete-time LQ problems, we could prove exponential decay of sensitivity, based on [Shin/Anitescu/Zavala '22]

Decaying sensitivity: Example

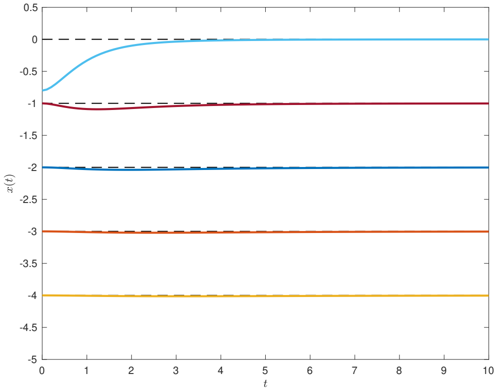


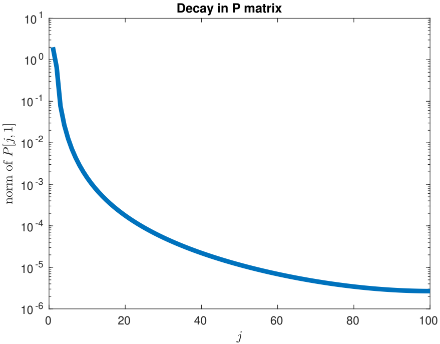


For discrete-time LQ problems, we could prove exponential decay of sensitivity, based on [Shin/Anitescu/Zavala '22]

It is not clear whether this also holds in this continuous-timeexample

Decaying sensitivity: Example





For discrete-time LQ problems, we could prove exponential decay of sensitivity, based on [Shin/Anitescu/Zavala '22]

It is not clear whether this also holds in this continuous-timeexample

Construction of overlapping approximation

Assume for sake of a simple presentation that the distance between z_i and z_j equals $|i - j|$

Construction of overlapping approximation

Assume for sake of a simple presentation that the distance between z_i and z_j equals $|i-j|$. Then the assumption on

$$z_k \mapsto V(z_1, \ldots, z_{l-1}, z_l, z_{l+1}, \ldots, z_s) - V(z_1, \ldots, z_{l-1}, 0, z_{l+1}, \ldots, z_s) \qquad (*)$$

implies

$$(*) \approx V(0, \ldots, 0, z_j, z_{j+1}, \ldots, z_{j+l}, 0, \ldots, 0) - V(0, \ldots, 0, 0, z_{j+1}, \ldots, z_{j+l}, 0, \ldots, 0)$$

UNIVERSITÄT BAYREUTH

Construction of overlapping approximation

Assume for sake of a simple presentation that the distance between z_i and z_j equals $|i-j|$. Then the assumption on

$$z_k \mapsto V(z_1,\dots,z_{l-1},z_l,z_{l+1},\dots,z_s) - V(z_1,\dots,z_{l-1},0,z_{l+1},\dots,z_s) \qquad (*)$$

implies

$$(*) \approx \underbrace{V(0,\dots,0,z_j,z_{j+1},\dots,z_{j+l},0,\dots,0) - V(0,\dots,0,0,z_{j+1},\dots,z_{j+l},0,\dots,0)}_{=:\ \Psi_l^j(z_j,\dots,z_{j+l})}$$

UNIVERSITÄT BAYREUTH

Construction of overlapping approximation

Assume for sake of a simple presentation that the distance between z_i and z_j equals $|i-j|$. Then the assumption on

$$z_k \mapsto V(z_1,\ldots,z_{l-1},z_l,z_{l+1},\ldots,z_s) - V(z_1,\ldots,z_{l-1},0,z_{l+1},\ldots,z_s) \qquad (*)$$

implies

$$(*) \approx \underbrace{V(0,\ldots,0,z_j,z_{j+1},\ldots,z_{j+l},0,\ldots,0) - V(0,\ldots,0,0,z_{j+1},\ldots,z_{j+l},0,\ldots,0)}_{=:\,\Psi_l^j(z_j,\ldots,z_{j+l})}$$

Then

$$V(x) \approx V(0) + \sum_{j=1}^{s} \Psi_l^j(z_j,\ldots,z_{j+l})$$

Construction of overlapping approximation

Assume for sake of a simple presentation that the distance between z_i and z_j equals $|i-j|$. Then the assumption on

$$z_k \mapsto V(z_1,\ldots,z_{l-1},z_l,z_{l+1},\ldots,z_s) - V(z_1,\ldots,z_{l-1},0,z_{l+1},\ldots,z_s) \qquad (*)$$

implies

$$(*) \approx \underbrace{V(0,\ldots,0,z_j,z_{j+1},\ldots,z_{j+l},0,\ldots,0) - V(0,\ldots,0,0,z_{j+1},\ldots,z_{j+l},0,\ldots,0)}_{=:\,\Psi_l^j(z_j,\ldots,z_{j+l})}$$

Then

$$V(x) \approx V(0) + \sum_{j=1}^{s} \Psi_l^j(z_j,\ldots,z_{j+l})$$

Concrete estimates in [Sperl/Saluzzi/Gr./Kalise '23] using exponentially decaying sensitivity, i.e., $|(*) - \psi_l^j| \le c\rho^j$ for some $\rho \in (0,1)$, yield

$$\left| V(x) - V(0) - \sum_{j=1}^{s} \Psi_l^j(z_j,\ldots,z_{j+l}) \right| \le c(s-1)\rho^j$$

Conclusions

- Deep neural networks can be used for computing Lyapunov functions, control Lyapunov functions, and approximations of optimal value function

Conclusions

- Deep neural networks can be used for computing Lyapunov functions, control Lyapunov functions, and approximations of optimal value function (this is not a new finding!)

Conclusions

- Deep neural networks can be used for computing Lyapunov functions, control Lyapunov functions, and approximations of optimal value function (this is not a new finding!)
- The method can overcome the curse of dimensionality if the approximated function has a compositional or — more specifically — separable structure

Conclusions

- Deep neural networks can be used for computing Lyapunov functions, control Lyapunov functions, and approximations of optimal value function (this is not a new finding!)
- The method can overcome the curse of dimensionality if the approximated function has a compositional or — more specifically — separable structure
- In this case we can handle dimensions that are far beyond those feasible for grid based methods

Conclusions

- Deep neural networks can be used for computing Lyapunov functions, control Lyapunov functions, and approximations of optimal value function (this is not a new finding!)
- The method can overcome the curse of dimensionality if the approximated function has a compositional or — more specifically — separable structure
- In this case we can handle dimensions that are far beyond those feasible for grid based methods
- Small-gain theory describes situations in which a compositional (control) Lyapunov function exists

Conclusions

- Deep neural networks can be used for computing Lyapunov functions, control Lyapunov functions, and approximations of optimal value function (this is not a new finding!)
- The method can overcome the curse of dimensionality if the approximated function has a compositional or — more specifically — separable structure
- In this case we can handle dimensions that are far beyond those feasible for grid based methods
- Small-gain theory describes situations in which a compositional (control) Lyapunov function exists
- Decaying sensitivity provides the existence of separable approximations to optimal value functions

Conclusions

- Deep neural networks can be used for computing Lyapunov functions, control Lyapunov functions, and approximations of optimal value function (this is not a new finding!)
- The method can overcome the curse of dimensionality if the approximated function has a compositional or — more specifically — separable structure
- In this case we can handle dimensions that are far beyond those feasible for grid based methods
- Small-gain theory describes situations in which a compositional (control) Lyapunov function exists
- Decaying sensitivity provides the existence of separable approximations to optimal value functions
- Topics of current and future research: separable approximate supersolutions, nonsmoothness, efficient training, relation to low-rank approximations

References

Lars Grüne, *Computing Lyapunov functions using deep neural networks*, Journal of Computational Dynamics 8 (2021), 131–152

Lars Grüne and Mario Sperl, *Examples for existence and non-existence of separable control Lyapunov functions*, Proceedings of NOLCOS 2022, IFAC-PapersOnLine 56 (2023), 19–24

Mario Sperl, Luca Saluzzi, Lars Grüne, and Dante Kalise, *Separable approximations of optimal value functions under a decaying sensitivity assumption*, arXiv 2304.06379, 2023

Kaiwen Chen and Alessandro Astolfi, *On the Active Nodes of Network Systems*, Proceedings of the 59th IEEE CDC, Jeju Island, Republic of Korea, 2020, 5561–5566

Wei Kang and Qi Gong, *Feedforward Neural Networks and Compositional Functions with Applications to Dynamical Systems*, SIAM Journal on Control and Optimization 60 (2022), 786–813,

Sungho Shin, Mihai Anitescu, and Victor M. Zavala, *Exponential decay of sensitivity in graph-structured nonlinear programs*, SIAM Journal on Optimization 32(2), (2023)