Physics-Informed and Data-Driven Learning of Lyapunov Functions with Formal Guarantees[†]

Jun Liu

Department of Applied Mathematics

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

ECC'24 Workshop on

Data-Driven Verification and Control with Provable Guarantees

Stockholm, June 25, 2024

[†]Joint work with Yiming Meng, Maxwell Fitzsimmons, Ruikun Zhou

Motivation



Stability, **safety**, and **performance** are often characterized by computationally challenging problems in systems and control:

- Stability and safety guarantees by **Lyapunov functions**.
- Optimal performance by **Hamilton–Jacobi–Bellman (HJB) equations**.

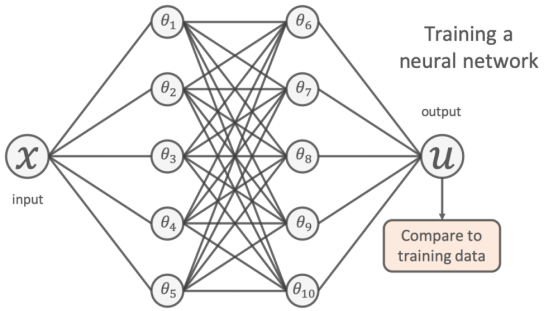
How to compute Lyapunov functions and solve HJB equations for high-dimensional systems remains a challenging task.

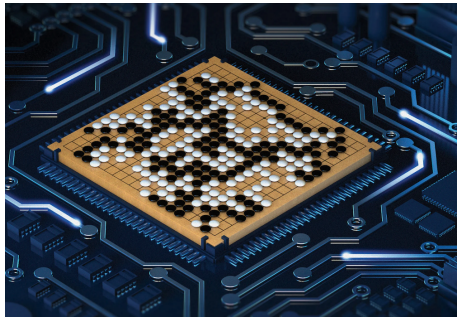
Neural networks are everywhere

Can they solve problems we cannot otherwise solve in **systems & control**?

Image created with DALL-E 3 via GPT-4.

Physics-informed neural networks





- Neural networks are known as universal function approximators and underpin the current revolution in AI.
- Many physics problems, including those in systems and control, are characterized by partial differential equations (PDEs).
- Physics-informed neural networks (PINNs) are specifically designed for solving PDEs.
- **Neural networks** are known to **lack formal guarantees** (e.g., CNNs misclassify, LLMs hallucinate, generative AIs ignore physics).

How to provide convergence, formal correctness, and performance guarantees of PINNs?

Outline

- Stability analysis using neural Lyapunov functions
- Optimal control using neural policy iteration
- Convergence of PINN approximations
- Formal verification of neural Lyapunov functions
- Implementations and examples

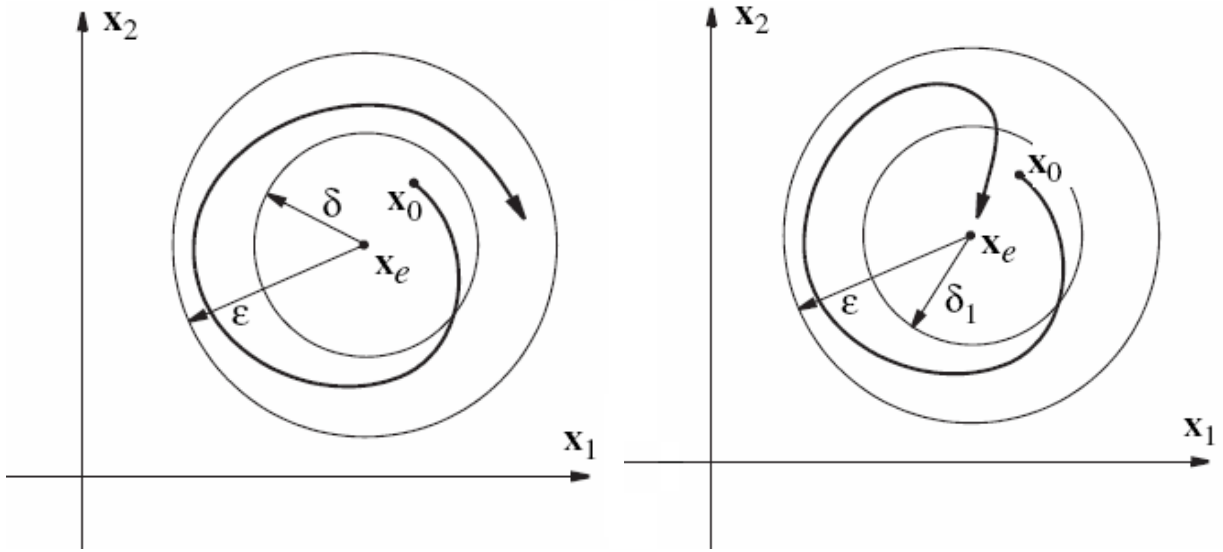
Stability analysis of an equilibrium point

- Consider a nonlinear system of the form

$$\dot{x} = f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a locally Lipschitz function. The unique solution to (1) with $x(0) = x_0$ is denoted by $\varphi(t, x_0)$.

- We assume that $x_e = 0$ is a locally **asymptotically stable** equilibrium point of (1).



- The **domain of attraction** of the origin for (1) is defined as

$$\mathcal{D} := \left\{ x \in \mathbb{R}^n : \lim_{t \to \infty} |\varphi(t, x)| = 0 \right\}. \tag{2}$$

Lyapunov Functions and regions of attraction

Lyapunov theorem (Lyapunov, 1892; Massera, Malkin, Kurzweil, 1950s)

The origin is asymptotically stable **if and only if** there exists a smooth function $V : D \to \mathbb{R}$ such that

$$\dot{V}(x) = \nabla V(x) \cdot f(x) < 0, \quad \forall x \in D \setminus \{0\}$$

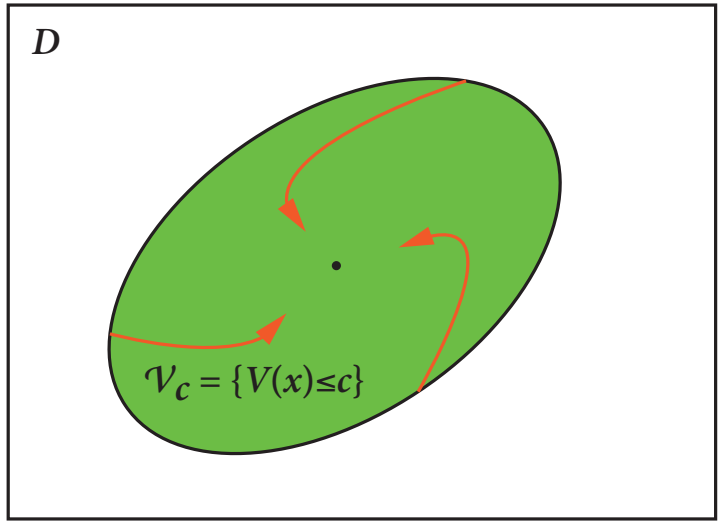
and

$$V(x) > 0, \quad \forall x \in D \setminus \{0\}.$$

- Any sub-level set of the form

  $$\mathcal{V}_c := \{x \in \mathbb{R}^n : V(x) \leq c\} \subseteq D$$

  is a **region of attraction**.
- Accurate estimates of ROA are crucial in applications.
- We can have $D = \mathcal{D}$.



Computing Lyapunov functions is a fundamental challenge

- Converse Lyapunov functions are usually nonconstructive.
- Despite its fundamental importance, computing Lyapunov functions is generally considered a challenging task.
- For linear systems $\dot{x} = Ax$, a quadratic Lyapunuv function $V(x) = x^T P x$ can be computed by solving the celebrated **Lyapunov equation**

$$PA + A^T P = -Q$$

  for any Hurwitz matrix $A \in \mathbb{R}^{n \times n}$ and positive definite symmetric matrix $Q \in \mathbb{R}^{n \times n}$.
- For polynomial systems, i.e., $f(x)$ is a polynomial function of x, semidefinite programming can be used to compute **sum-of-squares** (SOS) polynomial Lyapunov function.

A typical sum-of-squares approach

- A standard solution is to use sum-of-squares Lyapunov functions.
- **"Interior expanding" algorithm**[†]:

$$\begin{aligned}\{x \in \mathbb{R}^n \setminus \{0\} : p(x) \leq \beta\} &\subseteq \{x \in \mathbb{R}^n \setminus \{0\} : V(x) \leq \gamma\} \\ &\subseteq \{x \in \mathbb{R}^n : \nabla V(x) \cdot f(x) < 0\},\end{aligned}$$

  where V is an SOS polynomial. Can be formulated as a **bilinear** optimization from sum-of-squares relaxation.
- The choice of the "shape function" p can either be the Euclidean norm or heuristic[‡].
- Only works for polynomial systems. With polynomial approximations, formal guarantees are compromised.

[†] Packard, Andrew, et al. "Help on SOS." IEEE control systems 30.4 (2010): 18-23.

[†] Topcu, Ufuk, Andrew Packard, and Peter Seiler. "Local stability analysis using simulations and sum-of-squares programming." Automatica 44.10 (2008): 2669-2675.

[†] Khodadadi, Larissa, Behzad Samadi, and Hamid Khaloozadeh. "Estimation of region of attraction for polynomial nonlinear systems: A numerical method." ISA transactions 53.1 (2014): 25-32.

Typical neural network approaches

- Train a neural Lyapunov function $V(x) = V(x;\theta)$ with loss

$$\frac{1}{N}\sum_{i=1}^{N} c_1 \max(-V(x_i), 0) + c_2 \max(\nabla V(x_i)\cdot f(x_i), 0) + c_3 V(0)^2,$$

  plus other regularization or heuristic terms.
- Training is conducted on a set within the domain of attraction. Essentially a **local** approach.
- Satisfiability modulo theories (SMT) verification is often used to verify Lyapunov conditions and/or for counterexamples-guided synthesis.

Chang, Ya-Chien, Nima Roohi, and Sicun Gao. "Neural lyapunov control." NeurIPS (2019).

Grüne, Lars. "Computing Lyapunov functions using deep neural networks." Journal of Computational Dynamics 8.2 (2021): 131-152.

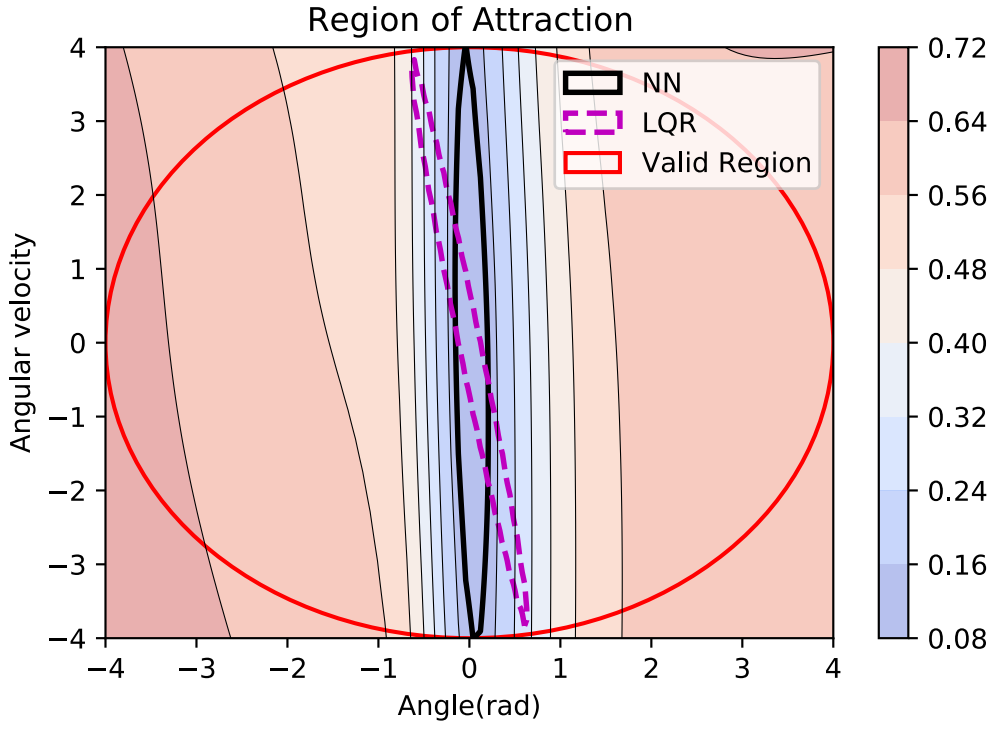
Abate, Alessandro, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. "Formal synthesis of Lyapunov neural networks." IEEE L-CSS, 2020.

Gaby, Nathan, Fumin Zhang, and Xiaojing Ye. "Lyapunov-Net: A deep neural network architecture for Lyapunov function approximation." 2022 IEEE CDC.

Zhou, Ruikun, Thanin Quartz, Hans De Sterck, and Jun Liu. "Neural Lyapunov control of unknown nonlinear systems with stability guarantees." NeurIPS (2022).

Typical neural network approaches

Comparisons are usually focused on the sizes of ROA **within** the domain of attraction.



(b) ROA comparison for inverted pendulum

Zhou, Ruikun, Thanin Quartz, Hans De Sterck, and Jun Liu. "Neural Lyapunov control of unknown nonlinear systems with stability guarantees." NeurIPS (2022).

Maximal Lyapunov function

- Let $\omega : \mathbb{R}^n \to \mathbb{R}$ be continuous and positive definite.
- Define

$$V(x) = \int_0^\infty \omega(\varphi(t,x))dt, \quad x \in \mathbb{R}^n, \tag{3}$$

  where, if the integral diverges, we let $V(x) = \infty$.
- Under mild technique assumptions, V has nice properties:
  1. $V(x) < \infty$ if and only if $x \in \mathcal{D}$;
  2. $V(x) \to \infty$ as $x \to y$ for some $y \in \partial\mathcal{D}$;
  3. V is positive definite on $\mathcal{D}$;
  4. V is continuous on $\mathcal{D}$ and its right-hand derivative along the solution of (1) satisfies

$$\dot{V}(x) := \lim_{t \to 0+} \frac{V(\varphi(t,x)) - V(x)}{t} = -\omega(x) \tag{4}$$

     for all $x \in \mathcal{D}$.
- V is a **maximal Lyapunov function**[†]. Rational functions were used by Vannelli & Vidyasagar (1985) to construct Lyapunov functions using an iterative procedure.

[†] Vannelli, Anthony, and Mathukumalli Vidyasagar. "Maximal Lyapunov functions and domains of attraction for autonomous nonlinear systems." Automatica, 1985.

A PDE characterization for Lyapunov functions

- The Lyapunov condition can be written as a partial differential equation (PDE):

$$\nabla V \cdot f = -\omega, \quad x \in \Omega, \tag{5}$$

  where Ω is an open set containing the origin.
- It is fair to call (5) the **Lyapunov equation**. It is a special case of a **Hamilton-Jacobi (HJ)** equation.
- *Special case*: $V(x) = x^T P x$, $\omega(x) = x^T Q x$, and $f(x) = Ax$.
- It is a linear first-order PDE. Solving this equation alone can lead to construction of Lyapunov functions with *remarkable* accuracy.
- If ω is locally Lipschitz, f is C^1, and $x = 0$ is exponentially stable, then V is locally Lipschitz on $\mathcal{D}$.
- If ω is C^1, f is C^1, and $x = 0$ is exponentially stable, then V is C^1 on $\mathcal{D}$.
- *Caveat*: Still require Ω to be contained in the domain of attraction: $V(x) \to \infty$ as $x \to \partial\mathcal{D}$.

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. "Physics-Informed Neural Network Lyapunov Functions: PDE Characterization, Learning, and Verification." arXiv:2312.09131 (2023) and provisionally accepted to Automatica

Zubov's theorem

- Let $\Omega \subseteq \mathbb{R}^n$ be an open set containing the origin. Then $\Omega = \mathcal{D}$ **if and only if** there exists two continuous functions $W : \Omega \to \mathbb{R}$ and $\Psi : \Omega \to \mathbb{R}$ such that the following conditions hold:
  1. $0 < W(x) < 1$ for all $x \in \Omega \setminus \{0\}$ and $W(0) = 0$;
  2. Ψ is positive definite on Ω with respect to the origin;
  3. for any sufficiently small $c_3 > 0$, there exist two positive real numbers c_1 and c_2 such that $|x| \geq c_3$ implies $W(x) > c_1$ and $\Psi(x) > c_2$;
  4. $W(x) \to 1$ as $x \to y$ for any $y \in \partial\Omega$;
  5. W and Ψ satisfy

$$\dot{W}(x) = -\Psi(x)(1 - W(x)), \tag{6}$$

  where $\dot{W}$ is the right-hand derivative of W along solutions of (1).
- We just transformed a function with range $[0, \infty)$ to $[0, 1]$.
- *Kruzkov transform*:

$$W(x) = 1 - \exp(-V(x)).$$

Used for modifying HJB equations to scale solutions. Lyapunov equation is an HJ equation.

A slightly more general PDE characterization

- Let $\beta : [0, \infty) \to \mathbb{R}$ satisfy

$$\dot{\beta} = (1 - \beta)\psi(\beta), \quad \beta(0) = 0, \tag{7}$$

  where ψ is a locally Lipschitz and $\psi(s) > 0$ for $s \geq 0$.
- A slightly generalized **Zubov equation**:

$$DW \cdot f = -\omega\psi(W)(1 - W), \tag{8}$$

  for $x \in \Omega$, where $\Omega \subseteq \mathbb{R}^n$ is an open set.
- Solutions to Lyapunov and Zubov equations are related by

$$W(x) = \begin{cases} \beta(V(x)), & \text{if } V(x) < \infty, \\ 1, & \text{otherwise,} \end{cases} \tag{9}$$

  where $\beta : [0, \infty) \to \mathbb{R}$ satisfies (7).
- *Special cases:* $\psi \equiv \alpha$ or $\psi(s) = \alpha(1 + s)$ for some constant $\alpha > 0$, which correspond to $\beta(s) = 1 - \exp(-\alpha s)$ and $\beta = \tanh(\alpha s)$, respectively.

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. "Physics-Informed Neural Network Lyapunov Functions: PDE Characterization, Learning, and Verification." arXiv:2312.09131 (2023) and provisionally accepted to Automatica

Main technical results

- We prove that there is a **unique viscosity solution** (Crandall & Lions, 1983) to Zubov's equation (8). This corresponds to the V we constructed (depending on solutions of the ODE).
- If ω is locally Lipschitz, f is C^1, and $x = 0$ is exponentially stable, then V is locally Lipschitz on $\mathbb{R}^n \setminus \partial\mathcal{D}$.
- If ω is C^1, f is C^1, and $x = 0$ is exponentially stable, then V is C^1 on $\mathbb{R}^n \setminus \partial\mathcal{D}$.
- **Error estimate:** For any $\delta > 0$, **ε-approximate** (i.e., ε-residual) viscosity solution with ε_b boundary error leads to

$$|V(x) - v(x)| \leq C(\varepsilon, \varepsilon_b)$$

  for all $x \in \bar{\Omega} \setminus \mathbb{B}_\delta$. We have $C(\varepsilon, \varepsilon_b) \to 0$ as $\varepsilon \to 0$ and $\varepsilon_b \to 0$.
- **Convergence:** A sequence of **ε-approximate viscosity solutions** $\{v_n\}$ with a uniform Lipschitz constant (or modulus of continuity) **uniformly converges** to V on $\bar{\Omega}$.

JL et al., Automatica'24

Physics-informed neural networks

- We can use neural networks to solve ODE/PDE, by minimizing the equation residual over a set of collocation points, while leveraging as much side information as possible.
- The idea recently gained remarkable interests as **physics-informed neural networks** (Lagaris *et al.*, 1998; Raissi *et al.*, 2019).
- Consider the generic first-order PDE

$$F(x, W, \nabla W) = 0, \quad x \in \Omega, \tag{10}$$

  subject to the boundary condition $W = g$ on $\partial\Omega$.
- A loss function for training a neural network solution $W_N(x; \theta)$ is defined as a weighted sum of the **equation residual** error, **initial/boundary** error, and **data** error (if available).

Lagaris, Isaac E., Aristidis Likas, and Dimitrios I. Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations." IEEE transactions on neural networks 9.5 (1998): 987-1000.

Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." Journal of Computational physics 378 (2019): 686-707.

Physics-informed neural network Lyapunov function

- For our purpose, the loss is given by

$$\begin{aligned}\text{Loss}(\theta) = \frac{1}{N_c}\sum_{i=1}^{N_c} \overbrace{F(x_i, W_N(x_i;\theta), \nabla W_N(x_i;\theta))^2}^{\text{PDE residual loss}} \\ + \lambda_b \frac{1}{N_b}\sum_{i=1}^{N_b} \overbrace{(W_N(y_i;\theta) - g(y_i))^2}^{\text{boundary loss}}, \\ + \lambda_d \frac{1}{N_d}\sum_{i=1}^{N_d} \overbrace{(W_N(z_i;\theta) - \hat{W}(z_i))^2}^{\text{data loss}}, \end{aligned} \tag{11}$$

  where F describes **Zubov's PDE** (8).
- We can generate data by solving the ODE (Wei *et al.*, 2023) and evaluate

$$V(x) = \int_0^\infty \omega(\varphi(t,x))dt, \quad W(x) = \begin{cases} \beta(V(x)), & \text{if } V(x) < \infty, \\ 1, & \text{otherwise.} \end{cases}$$

  See Meng et al. (2023) for a Koopman operator-based approach.

Formal verification of Lyapunov functions

- **Local stability**: linearization + second-order approximation to allow formal analysis near the origin.

$$x^T P x \leq c \Longrightarrow 2 \left\| P \cdot \nabla g(x) \right\| \leq r, \tag{12}$$

where $r < \lambda_{\min}(Q)$, $PA + A^T P = -Q$, $g(x) = f(x) - Ax$, by a mean value theorem argument.

- **Reachability:**

$$(c_1 \leq W_N(x) \leq c_2) \wedge (x \in X) \Longrightarrow \dot{W}_N(x) \leq -\varepsilon, \tag{13}$$

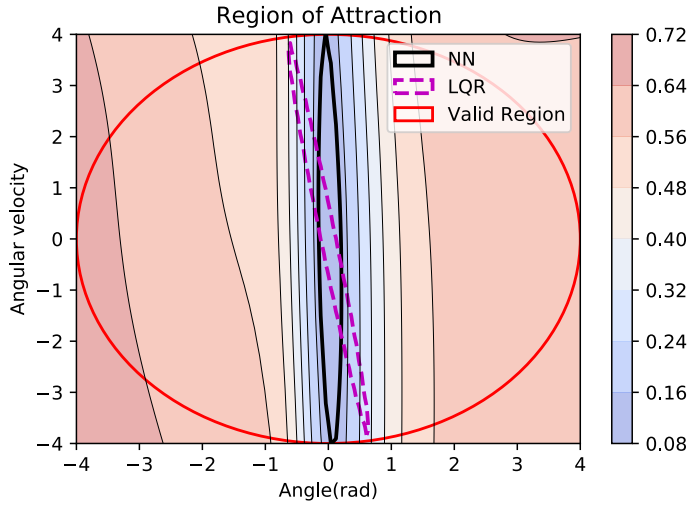
$$(W_N(x) \leq c_1) \wedge (x \in X) \Longrightarrow x^T P x \leq c, \tag{14}$$

where W_N is a neural Lyapunov function solving Zubov's PDE.

- We use the numerical SMT solver **dReal** (Gao, Kong, Clarke, 2013) to verify Lyapunov conditions are met by learned neural Lyapunov functions.
- Compositional verification can be done both locally and globally (if the system admits such a decomposition).

Numerical examples

ROA of a controlled inverted pendulum (NeurIPS 22):



(b) ROA comparison for inverted pendulum

Inverted pendulum with linear control:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -\sin(x_1) - x_2 - (k_1 x_1 + k_2 x_2), \end{aligned} \tag{15}$$

where the linear gains are given by $[k_1, k_2] = [4.4142, 2.3163]$.
Verification: With $Q = I$ and $r = 0.9999$, $2\|P \cdot Dg(x)\| \leq r$ can be verified with dReal within one millisecond, which confirms **global exponential stability** of the origin for (15).

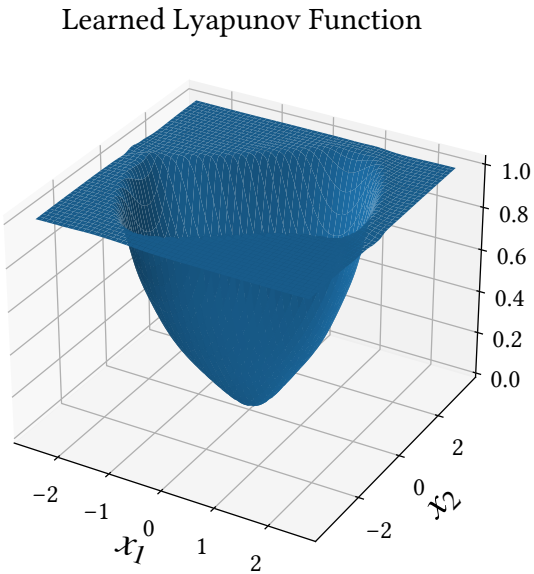
Numerical examples

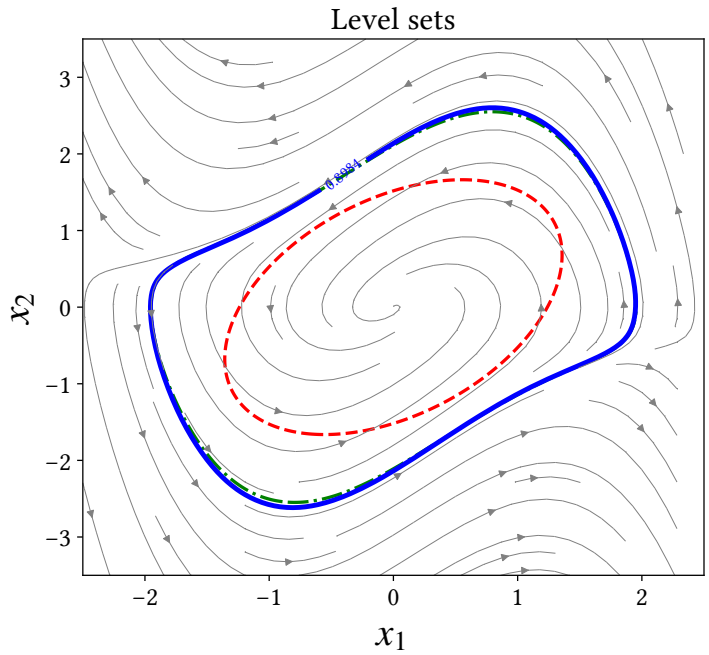
Reversed Van der Pol equation:

$$\begin{aligned} \dot{x}_1 &= -x_2, \\ \dot{x}_2 &= x_1 - \mu(1 - x_1^2)x_2, \end{aligned} \tag{16}$$

where $\mu > 0$ is a parameter that affects the stiffness of the equation.

- $\mu = 1$: verified level set compared with SOS





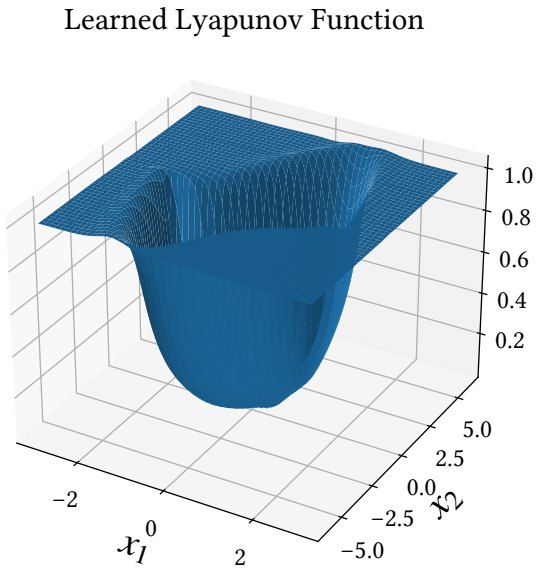
Numerical examples

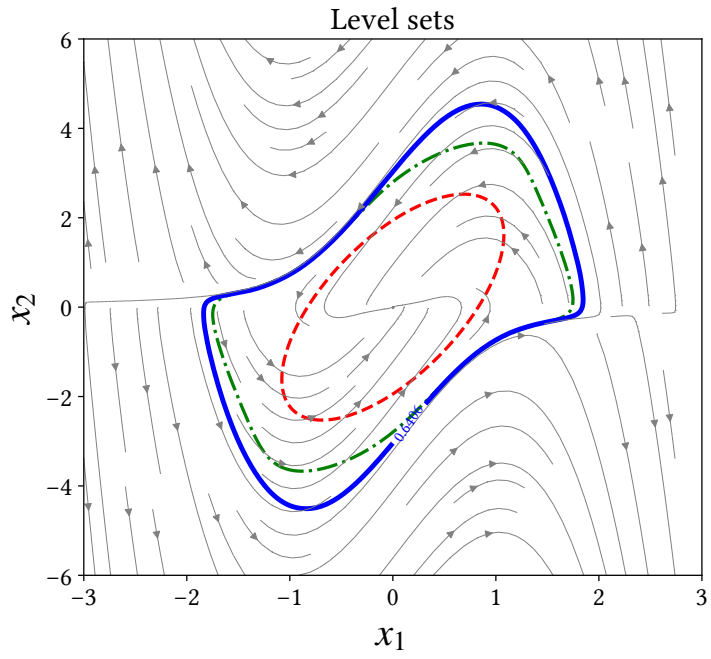
Reversed Van der Pol equation:

$$\begin{aligned} \dot{x}_1 &= -x_2, \\ \dot{x}_2 &= x_1 - \mu(1 - x_1^2)x_2, \end{aligned} \tag{17}$$

where $\mu > 0$ is a parameter that affects the stiffness of the equation.

- $\mu = 3$: verified level set compared with SOS





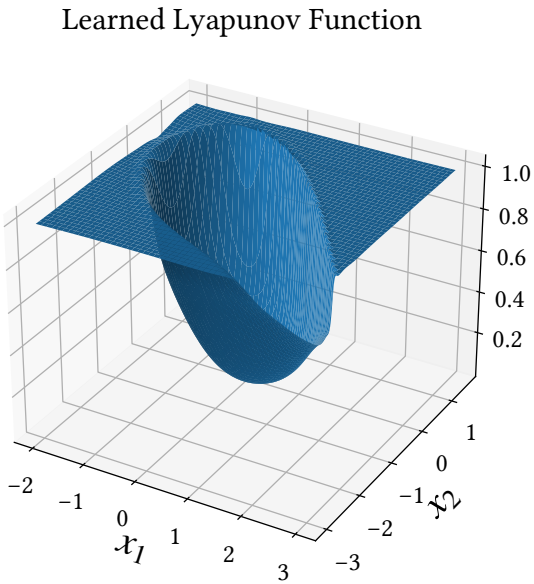
Numerical examples

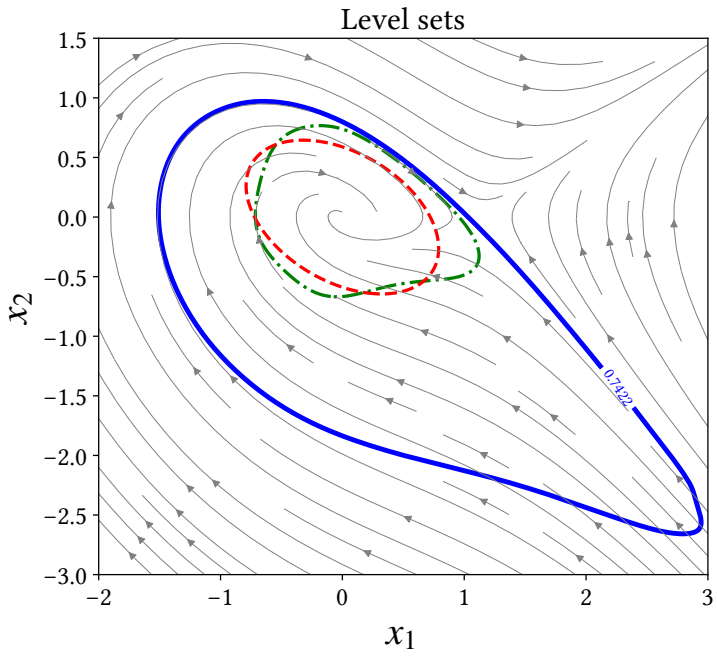
Two-machine power system (Vannelli & Vidyasagar):

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -0.5x_2 - (\sin(x_1 + \delta) - \sin(\delta)), \end{aligned} \tag{18}$$

where $\delta = \frac{\pi}{3}$. The system has an unstable equilibrium point at $(\pi/3, 0)$.

- Verified level set compared with SOS:





Data-driven learning with Zubov-Koopman operator

- Consider

$$v_t(x) := \int_0^t \omega(\varphi(r,x))dr \tag{19}$$

For any $h \in C_b(\mathbb{R}^n)$ and $t > 0$, define the **Zubov-Koopman operator** $\mathcal{T}_t : C_b(\mathbb{R}^n) \to C_b(\mathbb{R}^n)$ as

$$\mathcal{T}_t h(x) := \exp\{-v_t(x)\}\, h(\varphi(t,x)). \tag{20}$$

- We have $\lim_{k\to\infty} \mathcal{T}_{k\Delta} h = U$ for any $h \in C_b(\mathbb{R}^n)$ such that $h(0) = 1$ and any $\Delta > 0$, where $W = 1 - U$ solves Zubov's equation.
- We have developed a **data-driven algorithm** to approximate the operator $\mathcal{T}_\Delta$ as

$$\mathcal{T}_\Delta h = \hat{\mathcal{T}}\hat{h} + \theta,$$

where $\hat{h} \in \text{span}\{\psi_i\}_{i=1}^N$ corresponds to h,
$\hat{\mathcal{T}} : \text{span}\{\psi_i\}_{i=1}^N \to \text{span}\{\psi_i\}_{i=1}^N$, and θ is a residual term.

Meng *et al.*, TAC'24 (under review)

Data-driven learning with Zubov-Koopman operator

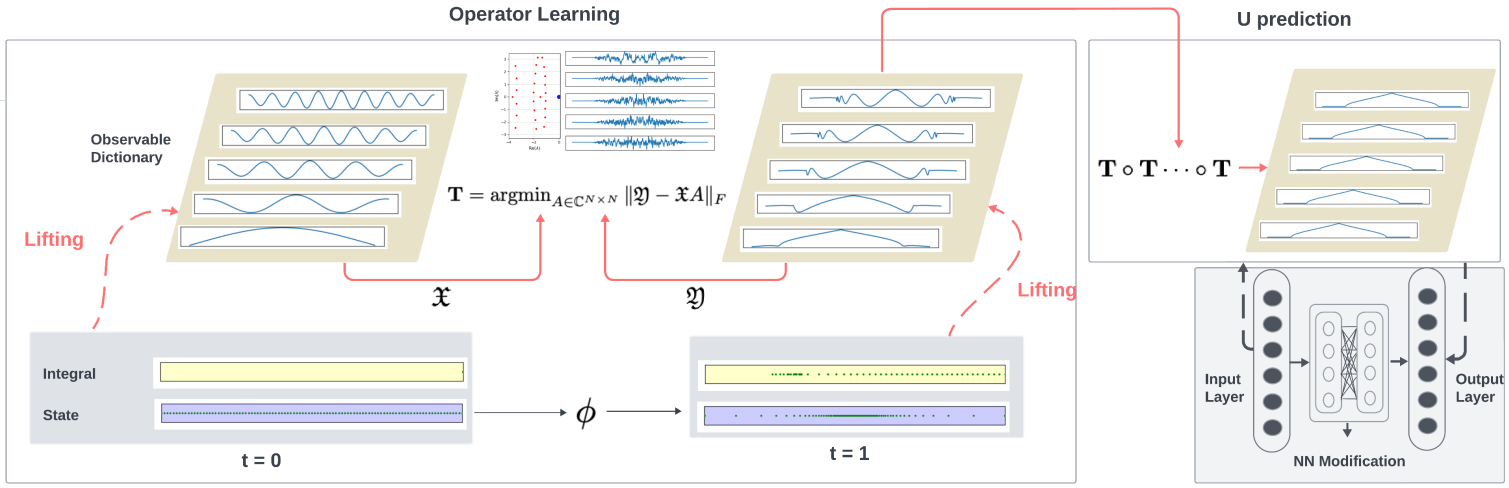
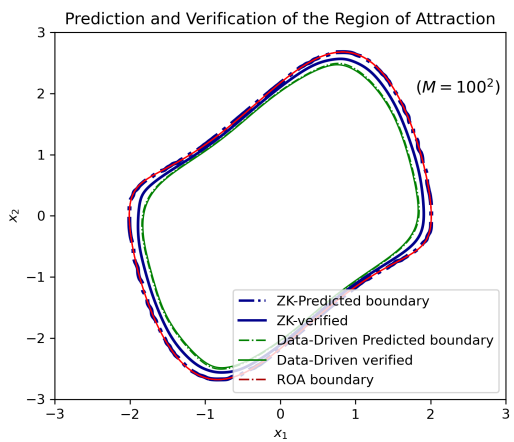
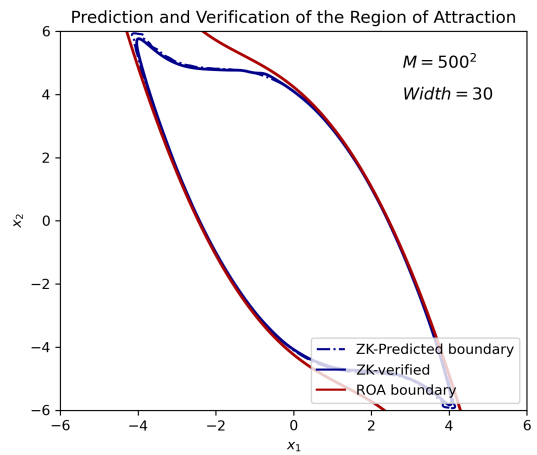
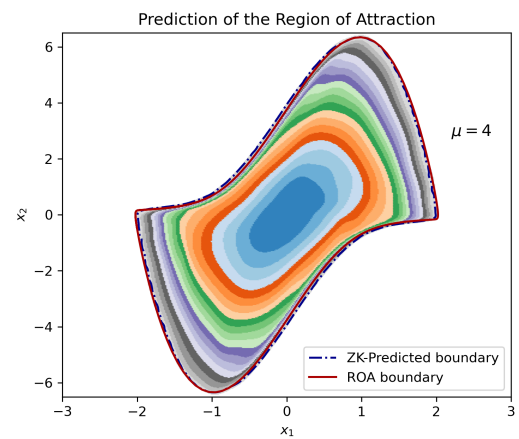


Fig. 1: Zubov-Koopman approach of ROA prediction and Lyapunov function construction.







Meng *et al.*, TAC'24 (under review)

Global neural Lyapunov functions for homogeneous systems

- Consider

$$\dot{\varphi} = f(\varphi), \quad \varphi(0, x) = x, \tag{21}$$

  where $f : \mathbb{R}^n \to \mathbb{R}^n$ is continuous, and $f(0) = 0$.
- Assume that f is **homogeneous** on $\mathbb{R}^n$ with degree d, i.e., for all real $\lambda > 0$, we have $f(\lambda x) = \lambda^d f(x)$ for all $x \in \mathbb{R}^n$.
- For a homogeneous system, Lyapunov conditions for local/global asymptotic stability reduce to

$$W(x) > 0, \tag{22}$$

$$\dot{W}(x) = \nabla W(x) \cdot f(x) < 0, \tag{23}$$

  for all $x \in S^{n-1}$, due to homogeneity of $\dot{W}$ and W.
- Locally asymptotically stable homogeneous vector fields admit global homogeneous Lyapunov functions (Hahn, 1967; Rosier, 1992). Relevance for general vector fields (Hermes, 1990s).
- We proposed a neural network architecture as a **universal approximation** of Lyapunov functions for homogeneous functions.

Numerical examples

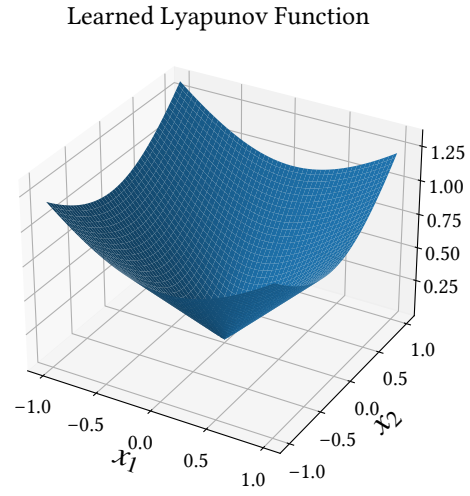
- Consider the homogeneous vector field:

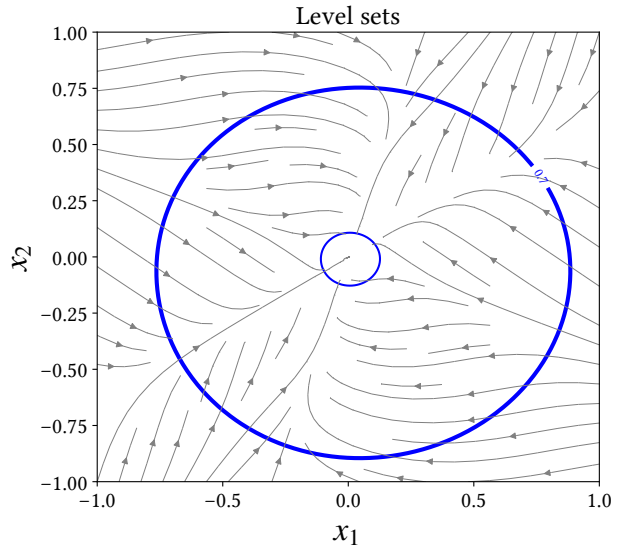
$$
\begin{aligned}
f_1 = &-\frac{1}{3}\Big(8x_1^{11}+22x_1^{10}x_2+93x_1^9x_2^2+126x_1^8x_2^3+483x_1^7x_2^4+186x_1^6x_2^5 \\
&+445x_1^5x_2^6-1189x_1^4x_2^7+234x_1^3x_2^8-713x_1^2x_2^9+1772x_1x_2^{10}-414x_2^{11}\Big) \\
&\times\sqrt{2x_1^6+8x_1^5x_2+22x_1^4x_2^2+18x_1^3x_2^3+26x_1^2x_2^4-16x_1x_2^5+13x_2^6} \\
&\Big/\left(2x_1^4-4x_1^3x_2+5x_1^2x_2^2+22x_1x_2^3+14x_2^4\right), \\
f_2 = &\frac{1}{3}\Big(6x_1^{11}+16x_1^{10}x_2+74x_1^9x_2^2+123x_1^8x_2^3+180x_1^7x_2^4-267x_1^6x_2^5 \\
&-1083x_1^5x_2^6-1729x_1^4x_2^7-1580x_1^3x_2^8+687x_1^2x_2^9+50x_1x_2^{10}-194x_2^{11}\Big) \\
&\times\sqrt{2x_1^6+8x_1^5x_2+22x_1^4x_2^2+18x_1^3x_2^3+26x_1^2x_2^4-16x_1x_2^5+13x_2^6} \\
&\Big/\left(2x_1^4-4x_1^3x_2+5x_1^2x_2^2+22x_1x_2^3+14x_2^4\right).
\end{aligned}
\tag{24}
$$

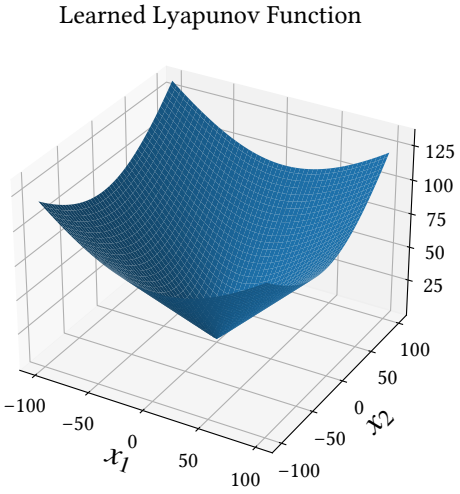
- The SOS approach is not applicable to this example since the vector field is not polynomial.

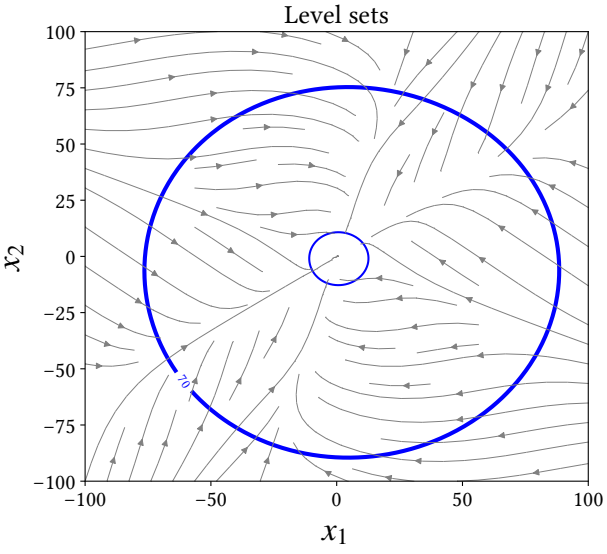
Numerical examples

- We use a one-layer neural network to prove global asymptotic stability of the homogeneous DE:









Stability analysis of interconnected systems

- Consider a network of nonlinear systems of the form

$$\dot{x}_i = f_i(x_i) + \sum_{j \neq i} G_{ij}(x_i, x_j), \tag{25}$$

  where each $f_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ and $G_{ij} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \to \mathbb{R}^{n_i}$ are assumed to be locally Lipschitz and $i, j \in \{1, \ldots, l\}$.
- Assumed that $f_i(0) = 0$ and $G_{ij}(0, 0) = 0$, i.e., the origin is an equilibrium point for each individual subsystem and for the overall interconnected system.
- We have developed an approach to compute **vector** neural Lyapunov functions by solving Zubov's equation using PINNs.
- **Composition verification** techniques can be used for efficient SMT verification of vector Lyapunov function conditions.

Liu *et al.*, ACC'24

A numerical example – networked Van der Pol oscillators

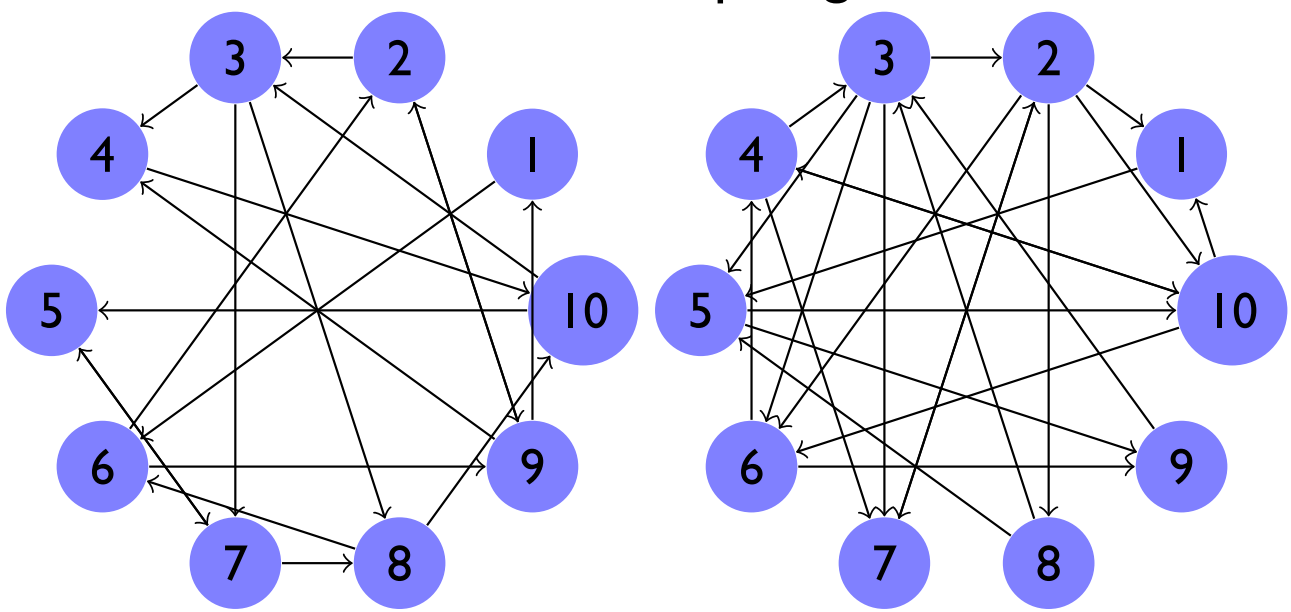
- Consider a **20-dimensional** system consisting of a network of reversed Van der Pol equations (Kundu and Anghel, 2015):

$$\dot{x}_{i1} = -x_{i2},$$

$$\dot{x}_{i2} = x_{i1} - \mu_i(1 - x_{i1}^2)x_{i2} + \sum_{j \neq i} \mu_{ij} x_{i1} x_{j2},$$

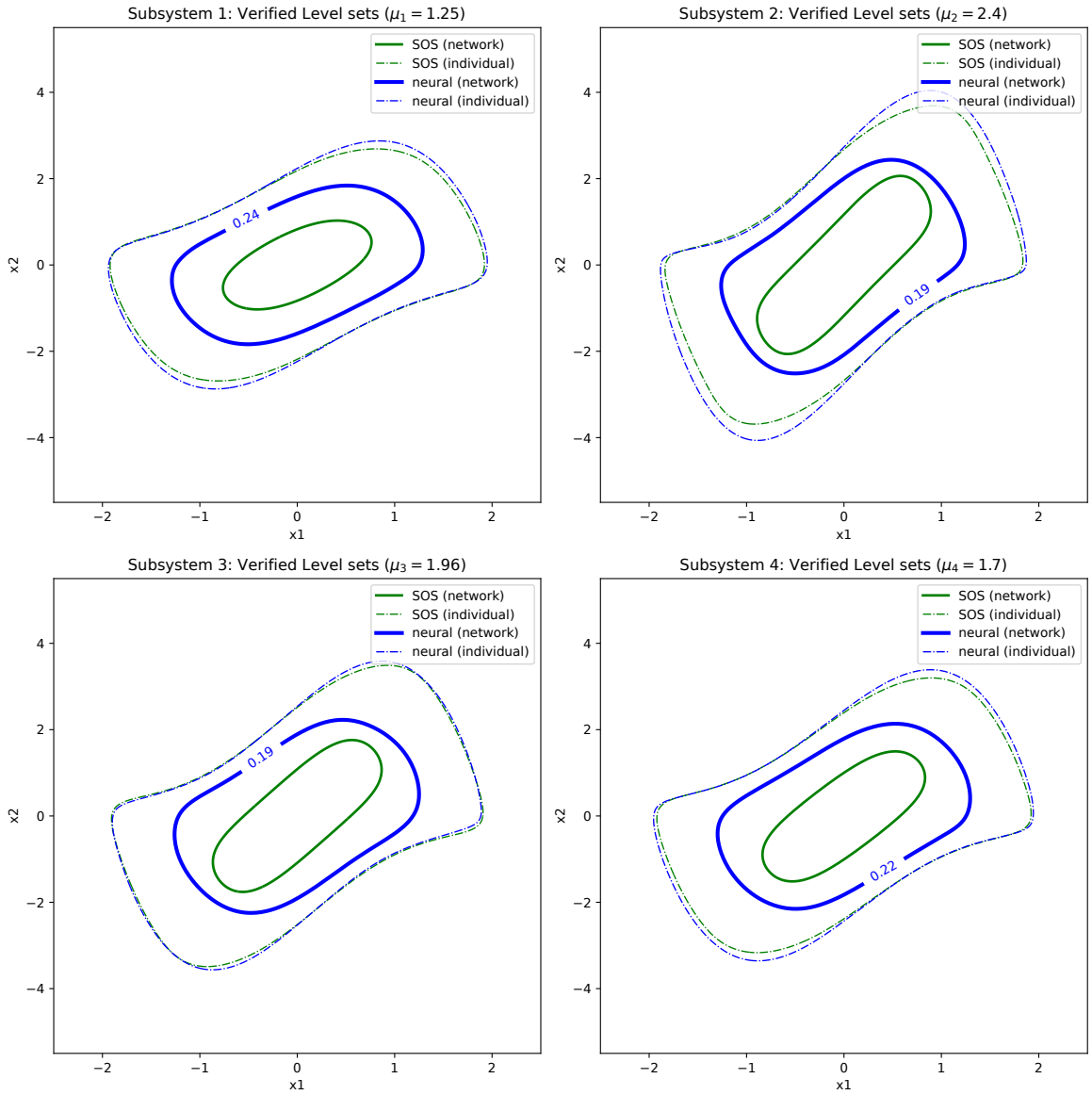
where $\mu_i \in (0.5, 2.5)$ and $\mu_{ij} \in (-0.1, 0.1)$ represents the interconnection strength.

- Consider two different network topologies:



A numerical example – networked Van der Pol oscillators

- Verified level sets (for subsystems 1–4) using **vector neural Lyapunov functions**, compared with SOS:



From stability analysis to optimal control

- Consider a control-affine system

$$\dot{x} = f(x) + g(x)u, \tag{26}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a continuously differentiable vector field and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ is smooth, $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input. We also assume that $f(0) = 0$.

- Consider a cost

$$J(x, u) = \int_0^\infty Q(x) + u^T R(x) u ds, \tag{27}$$

where $Q(x)$ and $R(x)$ are positive definite for all x.

- A control $u : \Omega \to \mathbb{R}^m$ is said to be admissible, if (1) u is locally Lipschitz; (2) $u(0) = 0$; (3) u is a stabilizing control, i.e., $\lim_{t\to\infty} |\varphi(t; x_0, u)| = 0$ for all $x_0 \in \Omega$; and (4) $J(x_0, u) < \infty$.
- Denote the set of admissible controls by $\mathcal{U}(\Omega)$. The goal is to compute u^* such that $J(x, u^*) = \inf_{u\in\mathcal{U}(\Omega)} J(x, u), \quad \forall x \in \Omega$.

Generalized HJB and policy iteration

- The Hamilton-Jacobi-Bellman (HJB) equation is

$$-Q(x)-\nabla V(x)\cdot f(x)+\frac{1}{4}\nabla V(x)g(x)R^{-1}g^T(x)(\nabla V(x))^T=0. \quad (28)$$

- A classical algorithm of policy iteration (Vaisbord, 1963; Milshtein, 1964; Leake and Liu, 1967; Saridis and Lee, 1979) solves the HJB equation (28) by iteratively solving a linear Lyapunov-type PDE:

$$Q(x)+\kappa_i^T(x)R(x)\kappa_i(x)+\nabla V_i(x)(f(x)+g(x)\kappa_i(x))=0, \quad (29)$$

  and updating

$$\kappa_{i+1}(x)=-\frac{1}{2}R^{-1}g^T(x)(\nabla V_i(x))^T. \quad (30)$$

- Beard (1995) termed (29) the generalized HJB and investigated Galerkin method for solving it. See also Jiang, Lewis, and coauthors for data-driven variants of this approach.

Convergence of exact policy iteration to viscosity solutions

- We prove that policy iteration indeed converges to viscosity solutions (Crandall & Lions, 1983) of the HJB equation.
- (**Exact PI**) Suppose that $u_0 \in \mathcal{U}(\Omega)$, then,
  (1) $u_i \in \mathcal{U}(\Omega)$ for all $i \in \{0, 1, \cdots\}$.
  (2) $V^* \leq V_{i+1} \leq V_i$ for all $x \in \Omega$ and for all $i \in \{0, 1, \cdots\}$, where V_i is the viscosity solution to $G(x, \kappa_i(x), DV_i(x)) = 0$ and V^* is the viscosity solution to the HJB equation (28).
  (3) $V_i \to V^*$ uniformly on Ω as $i \to \infty$ given the compactness of Ω.
- Previous work requires stronger regularity assumptions on V.

Meng, Yiming, Ruikun Zhou, Amartya Mukherjee, Maxwell Fitzsimmons, Christopher Song, and Jun Liu. "Physics-Informed Neural Network Policy Iteration: Algorithms, Convergence, and Verification." arXiv preprint arXiv:2402.10119 (2024), to appear in ICML'24.

Convergence of neural policy iteration to viscosity solutions

- (**Lyapunov PDE**) Assume that the training error $E_{T,N}(\hat{V}_N)$ can be arbitrarily small for sufficiently large N. Then the neural network $\hat{V}_N \to V$ in $\mathcal{G}$, provided that the sequence of approximations $\left\{\hat{V}_N\right\}$ lies in a compact subset of $\mathcal{G} = C^1(\Omega \setminus U_\varepsilon, \mathbb{R})$ equipped with the C^1-uniform norm, $\|V\|_{C^1} := \sup_{x\in\Omega\setminus U_\varepsilon} \|V(x)\| + \sup_{x\in\Omega\setminus U_\varepsilon} \|\nabla V(x)\|$.
- (**Neural PI**) By patching the approximation on $\Omega \setminus U_\varepsilon$ for any small $\varepsilon > 0$ and the asymptotic approximation within U_ε (boundary condition),
for any $i \geq 0$ and $\vartheta > 0$, we can choose a sufficiently dense set of collocation points $\{x_k\}_{k=1}^N$ such that

$$|\hat{V}_i(x) - V_i(x)| \leq \vartheta, \ |\hat{\kappa}_{i+1}(x) - \kappa_{i+1}(x)| \leq \vartheta, \quad x \in \Omega.$$

Meng, Yiming, Ruikun Zhou, Amartya Mukherjee, Maxwell Fitzsimmons, Christopher Song, and Jun Liu. "Physics-Informed Neural Network Policy Iteration: Algorithms, Convergence, and Verification." arXiv preprint arXiv:2402.10119 (2024), to appear in ICML'24.

Extreme learning machine policy iteration

Algorithm 1 Extreme Learning Machine Policy Iteration (ELM-PI)

Require: f, g, Q, R, k_0, Ω, N, m

1: **repeat**
2: Generate random W and b
3: Generate random $\{x_s\}_{s=1}^{N} \subseteq \Omega$
4: Finding β that minimizes (14) to form V_i from (11)
5: Update κ_{i+1} according to (10)
6: $i = i + 1$
7: **until** desired accuracy or max iterations reached

- **ELM-PI** uses linear least squares to optimize β in

$$V_i(x;\beta) = \beta^T \sigma(Wx + b),$$

to minimize the loss

$$\text{Loss}(\beta) = \frac{1}{N}\sum_{s=1}^{N} H(x_s, \beta^T \text{diag}(\sigma'(Wx+b))W)^2 + \lambda(\beta^T\sigma(b))^2,$$

where H is the right-hand side of the GHJB at each iteration.

Ruikun Zhou *et al.*, CSS-L; Yiming Meng, Ruikui Zhou, *et al.*, ICML'24.

Physics-informed neural network policy iteration

Algorithm 2 Physics-Informed Neural Network Policy Iteration (PINN-PI)

Require: f, g, Q, R, k_0, Ω, $V_{i,\mathrm{NN}}(x;\theta)$

repeat
  Generate random $\{x_s\}_{s=1}^{N} \subseteq \Omega$
  repeat
    Run gradient descent on θ with (16)
  until desired accuracy or max epochs reached
  Form $V_i(x)$ from $V_{i,\mathrm{NN}}(x;\theta)$
  Update κ_{i+1} according to (10)
  $i = i + 1$
until desired accuracy or max iterations reached

- **PINN-PI** uses gradient descent to optimize θ in $V_{i,\mathrm{NN}}(x;\theta)$ by minimizing the loss

$$\mathrm{Loss}(\theta) = \frac{1}{N}\sum_{s=1}^{N} H(x_s, \nabla V_{i,\mathrm{NN}}(x_s;\theta))^2 + \lambda (V_{i,\mathrm{NN}}(0;\theta))^2,$$

where $\lambda > 0$ is a weight parameter and H is the right-hand side of the GHJB at each iteration.

Yiming Meng, Ruikui Zhou, *et al.*, ICML'24.

Numerical examples

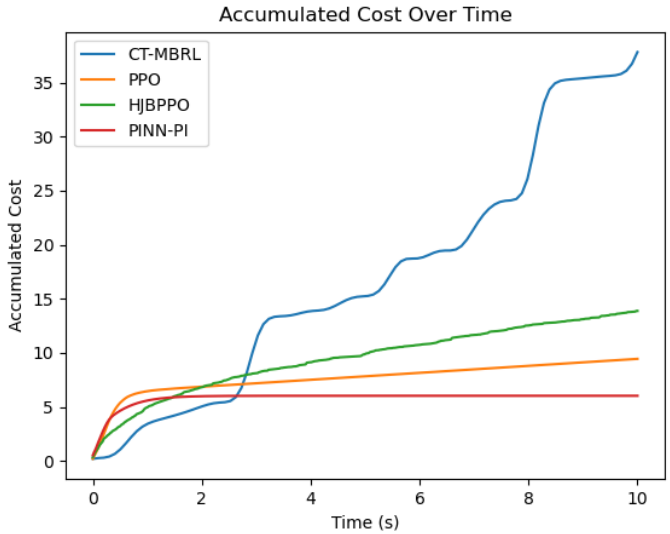
- We conducted comparisons with successive Galerkin approximations (SGA) and popular RL algorithms (PPO, CT-MBRL, and HJBPPO) on benchmark control problems (inverted pendulum, cartpole, 2D quadrotor, 3D quadrotor).
- ELM-PI excels in low-dimensional systems and achieves remarkable accuracy.

SGA			ELM-PI			PINN-PI		
Order	Time (s)	Verified?	m	Time (s)	Verified?	m	Time (s)	Verified?
2	4.80	Yes	50	**0.11**	Yes	50	255.15	Yes
4	19.37	Yes	100	**0.24**	Yes	100	256.53	Yes
6	66.52	Yes	200	**0.71**	Yes	200	258.89	Yes
8	212.42	Yes	400	**2.92**	Yes	400	256.52	Yes

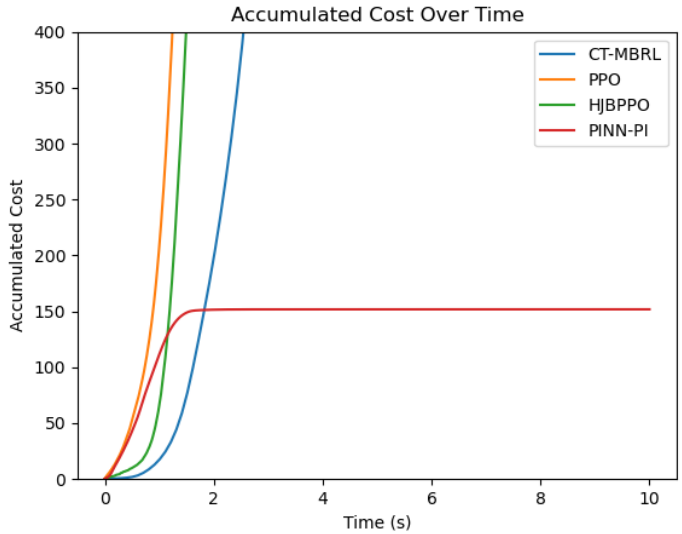
- PINN-PI performs well in high-dimensional systems and outperforms both SGA and RL algorithms.

Ruikun Zhou *et al.*, CSS-L; Yiming Meng *et al.*, ICML'24.

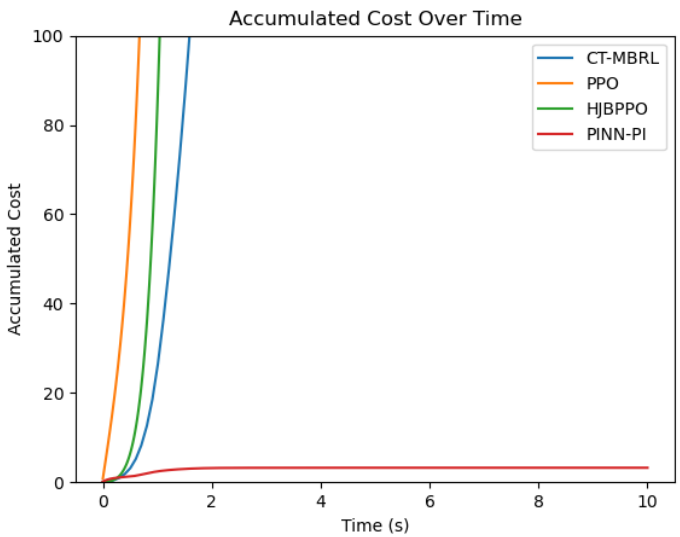
Numerical examples – comparison of accumulated costs



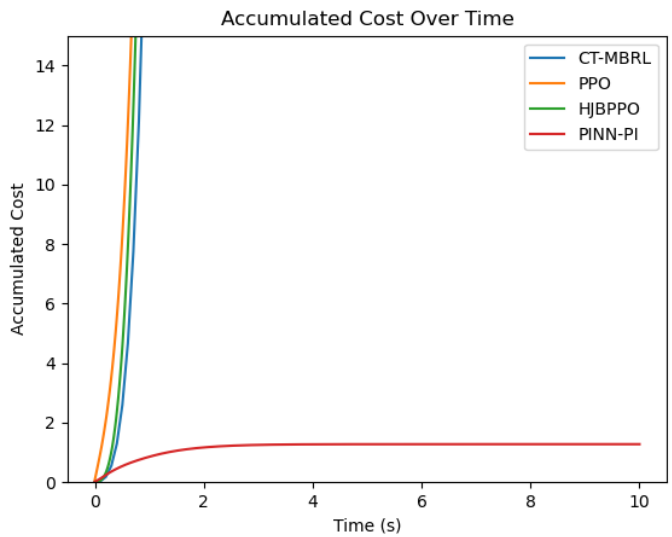
(a) Inverted Pendulum



(b) Cartpole

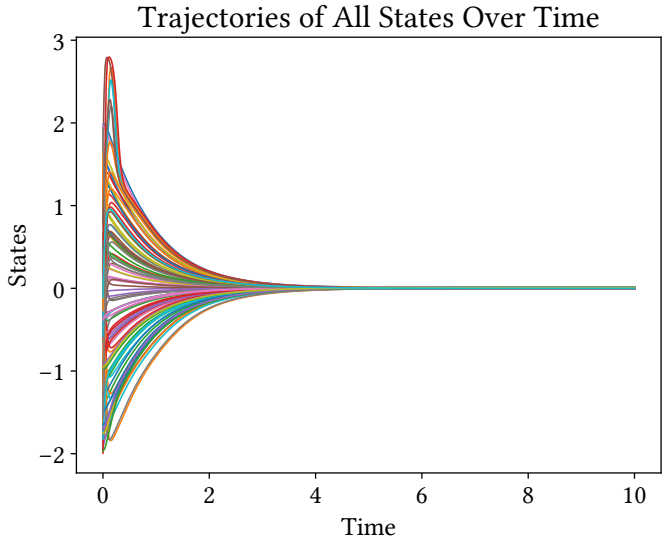


(c) 2D Quadrotor

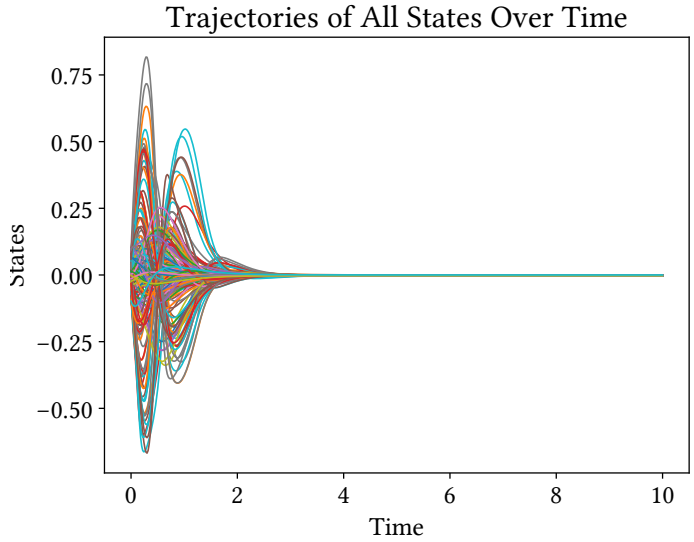


(d) 3D Quadrotor

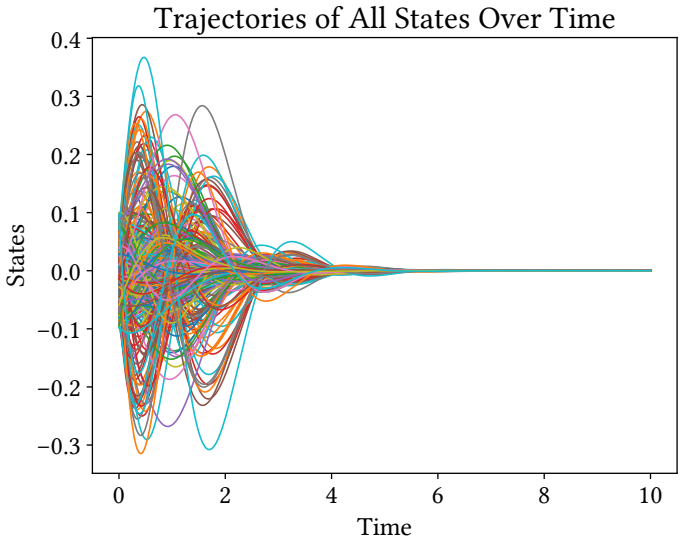
Numerical examples – simulated closed-loop trajectories



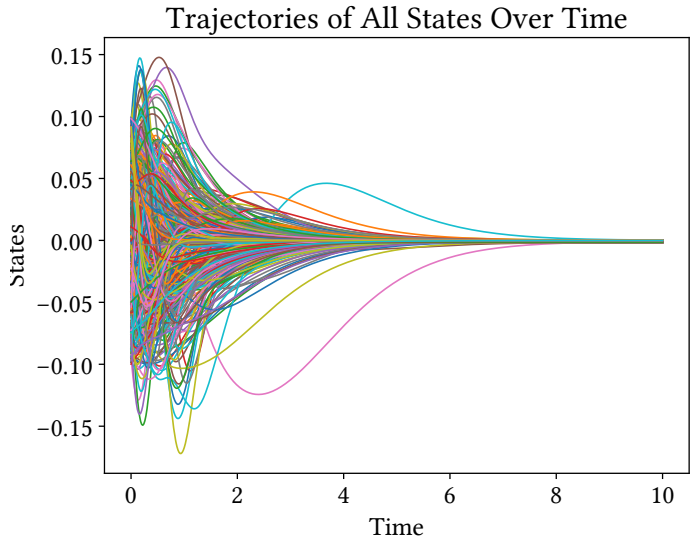
(a) Inverted Pendulum



(b) Cartpole



(c) 2D Quadrotor



(d) 3D Quadrotor

LyZNet – A lightweight toolbox for learning and verification of neural Lyapunov functions

- An integrated set of Python modules built upon PyTorch and SMT solvers including dReal (Gao, 2013). Leverages NumPy, SciPy, and SymPy (and dReal) for numerical, scientific, and symbolic computation.
- Support different classes of dynamical systems (e.g., nonlinear, control-affine, interconnected, homogeneous, stochastic).
- Allow various neural network models (nonlinear, polynomial, homogeneous) and loss functions (Lyapunov, Zubov, HJB, GHJB)
- Support different training/optimization algorithms (gradient descent, linear least squares, SDP).
- Support different verification engines (dReal, Z3).

https://git.uwaterloo.ca/hybrid-systems-lab/lyznet

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. "LyZNet: A Lightweight Python Tool for Learning and Verifying Neural Lyapunov Functions and Regions of Attraction." In Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control, 2024.

LyZNet – A minimal example

pendulum_elm_pi.py

```python
import sympy as sp
import lyznet

lyznet.utils.set_random_seed(123)

x1, x2 = sp.symbols('x1 x2')
symbolic_vars = (x1, x2)

f = sp.Matrix([x2, 19.6*sp.sin(x1) - 4.0*x2])
g = sp.Matrix([0, 40.0])

domain = [[-2, 2]] * 2
sys_name = f"pendulum_elm_pi_{domain[0]}"

R = 2.0*sp.eye(1)

initial_u = sp.Matrix([-(1/40)*x1 - (19.6/40)*sp.sin(x1)])

system = lyznet.ControlAffineSystem(f, g, domain, sys_name, R=R)

lyznet.elm_pi(system, initial_u=initial_u, num_of_iters=10, width=400,
              num_colloc_pts=3000, final_plot=True)

```

```
------------------------------------------------------------
Total time for EML training: 1.25 seconds.
Plotting learned Lyapunov function and level sets...
Elapsed time: 1.3124 seconds.
Simulating 50 trajectories from random intitial conditions...
Elapsed time: 2.4766 seconds.
```

Summary

- Learn neural Lyapunov/value functions with PINNs.
- Verify neural Lyapunov functions with SMT solvers.
- Theoretical analysis of error estimates and convergence.
- Promising results compared with standard SOS approaches.
- Extensions to global stability analysis of homogeneous systems.
- Extensions to compositional verification for interconnected systems.
- An integrated learning and verification tool (LyZNet).

Future work

- Automatic discovery of compositional structure.
- Learn to stabilize (without an initial controller).
- Extensions to safe stabilization, reach-avoid-stay, and other complex specifications.

References

PINN Lyapunov functions:

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. "Physics-Informed Neural Network Lyapunov Functions: PDE Characterization, Learning, and Verification." arXiv:2312.09131 (2023), provisionally accepted to Automatica.

Optimal control:

Meng, Yiming, Ruikun Zhou, Amartya Mukherjee, Maxwell Fitzsimmons, Christopher Song, and Jun Liu. "Physics-Informed Neural Network Policy Iteration: Algorithms, Convergence, and Verification." arXiv preprint arXiv:2402.10119, ICML 2024.

Data-driven Zubov-Koopman lifting:

Meng, Yiming, Ruikun Zhou, and Jun Liu. "Learning Regions of Attraction in Unknown Dynamical Systems via Zubov-Koopman Lifting: Regularities and Convergence." arXiv:2311.15119 (2023).

LyZNet tool:

Liu, Jun, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. "TOOL LyZNet: A Lightweight Python Tool for Learning and Verifying Neural Lyapunov Functions and Regions of Attraction." In Proc. of HSCC, 2024.

Homogeneous DEs (L-CSS'24); **Compositional verification** (ACC'24)