ENFORCING ROBUST CONTROL GUARANTEES WITHIN NEURAL NETWORK POLICIES

3/11/22

A Paper By: Donti et al. [1]

Presentation By: Pouya Kananian,
Department of Electrical and Computer Engineering

Course: CS 885 - Instructor: Prof. Pascal Poupart

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Photo credit: @bruce.digital

INTRODUCTION

Robust Control

- The field of robust control has been able to provide rigorous guarantees on when controllers will succeed or fail in controlling a system of interest.
- If the uncertainties in the underlying dynamics can be bounded in specific ways, these techniques can produce controllers that are provably robust even under worst-case conditions.
- However, as the resulting policies tend to be simple (i.e., often linear).
- In contrast, deep reinforcement learning models are able to capture complex, nonlinear model.
- However, due to a lack of robustness guarantees, these techniques have still found limited application in safety-critical domains.

Combining Robust Control and Deep RL

- This paper proposes a method for combining the guarantees of robust control with the flexibility of deep reinforcement learning.
- We consider the setting of nonlinear, time-varying systems with unknown dynamics, but the uncertainty on these dynamics can be bounded
- Building upon specifications provided by traditional robust control methods in these settings, we construct a new class of nonlinear policies that are parameterized by neural networks, but that are nonetheless *provably robust*
- We *project* the outputs of a nominal (deep neural network-based) controller onto a space of stabilizing actions characterized by the robust control specifications

Addressing the lack of safety and stability in RL

- Combine control-theoretic ideas, predominantly robust control, with the nonlinear control policy benefits of RL.
- Safe RL
  - Learning control policies while maintaining some notion of safety during or after learning.
  - Typically, these methods attempt to restrict the RL algorithm to a safe region of the state space by making strong assumptions about the smoothness of the underlying dynamics.
  - This framework is in theory more general than our approach, which requires using stringent uncertainty bounds

BACKGROUND

Linear Matrix Inequalities

- In convex optimization, a linear matrix inequality (LMI) is an expression of the following form:

$$LMI(x) := A_0 + \sum_{i=0}^{m} x_i A_i \geq 0$$

- Robust control is concerned with the design of feedback controllers with guaranteed performance under worst-case conditions.
- Many classes of robust control problems in both the time and frequency domains can be formulated using linear matrix inequalities (LMIs).
- Providing stability guarantees often requires the use of simple (linear) controllers, which greatly limits average-case performance

Linear Differential Inclusions

- Our aim is to control nonlinear (continuous-time) dynamical systems of the form

$$\dot{x}(t) \in A(t)x(t) + B(t)u(t) + G(t)w(t)$$

x(t): State at time t

u(t): Control input

w(t): Captures both external disturbance and any modeling discrepancies

- This class of models is referred to as linear differential inclusions (LDIs)
- Despite the name: Can characterize nonlinear systems
- Within this class of models, it is often possible to construct robust control specifications certifying system stability

Robust Control Specifications

Our system: $\dot{x}(t) \in A(t)x(t) + B(t)u(t) + G(t)w(t)$

- In the continuous-time, infinite-horizon settings, the goal is often to construct a time-invariant control policy $\mathrm{u(t)} = \pi(x(t))$
- Alongside constructing some certification that guarantees stability.
- For many systems, this certification is in the form of a PD Lyapunov function.

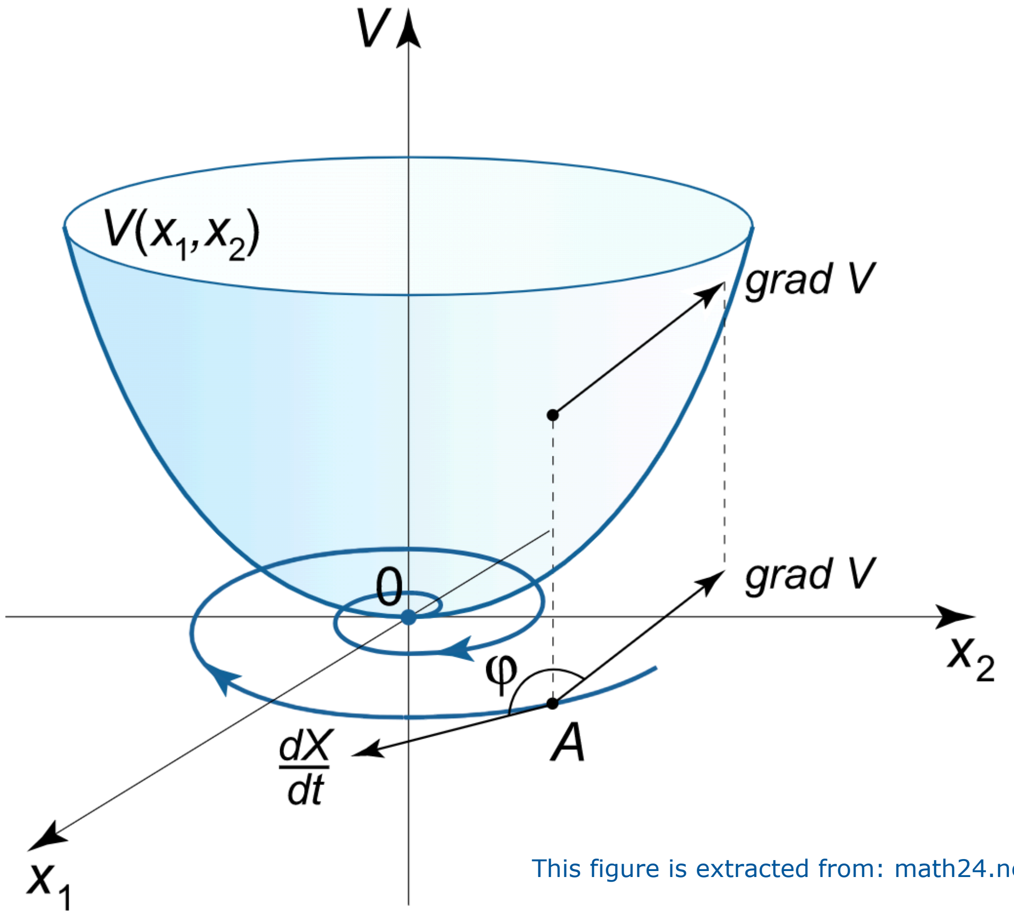
$$V\colon \mathbb{R}^s \to \mathbb{R}, \mathrm{V}(0) = 0, \mathrm{V}(\mathrm{x}) > 0 \text{ for all } \mathrm{x} \neq 0$$

$$\dot{V}\big(x(t)\big) \leq -\alpha \mathrm{V}(\mathrm{x}(\mathrm{t})), \text{ for some } \alpha > 0$$

Safety Guarantees

- $V\colon \mathbb{R}^s \rightarrow \mathbb{R}, \mathrm{V(0) = 0, V(x) > 0 \text{ for all } x \neq 0}$
- $\dot{V}\big(x(t)\big) \leq -\alpha \mathrm{V(x(t))}, \text{ for some } \alpha > 0$

- $\dot{V}\big(x(t)\big) \leq 0$
- $\dot{V}\big(x(t)\big) \leq -\alpha \mathrm{V(x(t))}, \text{ for some } \alpha > 0$



This figure is extracted from: math24.net[3]

Robust Control Specifications

Our system: $\dot{x}(t) \in A(t)x(t) + B(t)u(t) + G(t)w(t)$

Lyapunov function: $V\colon \mathbb{R}^s \to \mathbb{R}, \mathrm{V}(0) = 0, \mathrm{V}(\mathrm{x}) > 0 \text{ for all } \mathrm{x} \neq 0$

$$\dot{V}\big(x(t)\big) \leq -\alpha \mathrm{V}(\mathrm{x}(\mathrm{t})), \text{ for some } \alpha > 0$$

- For certain classes of bounded dynamical systems, it is possible to construct safety guarantees using semidefinite programming
  - time-invariant linear control policies $u(t) = Kx(t)$
  - and quadratic Lyapunov functions $V(x) = x^T Px$
- For instance, consider the class of norm-bounded LDIs (NLDIs)

$$\underline{\dot{x} = Ax(t) + Bu(t) + Gw(t)}, \underline{\|w(t)\|_2 \leq \|Cx(t) + Du(t)\|_2}$$

Robust Control Specifications

Our system: $\dot{x} = Ax(t) + Bu(t) + Gw(t), \|w(t)\|_2 \leq \|Cx(t) + Du(t)\|_2$

Safety Specifications: $V\colon \mathbb{R}^s \to \mathbb{R}, \mathrm{V}(0) = 0, \mathrm{V(x)} > 0 \text{ for all } \mathrm{x} \neq 0$

$\dot{V}\big(x(t)\big) \leq -\alpha \mathrm{V(x(t))}, \text{ for some } \alpha > 0$

- For these systems, it is possible to specify a set of stabilizing policies via a set of linear matrix inequalities

$$\begin{bmatrix} AS + SA^T + \mu GG^T + BY + Y^T B^T + \alpha S & SC^T + Y^T D^T \\ CS + DY & -\mu I \end{bmatrix} \preceq 0, \quad S \succ 0, \quad \mu > 0,$$

- For matrices S and Y satisfying the above inequality, $K = YS^{-1}$ and $P = S^{-1}$ are then a stabilizing linear controller gain and Lyapunov matrix, respectively.

Control Objectives

- To make comparisons with existing methods, we consider the infinite-horizon "linear-quadratic regulator" (LQR) cost:

$$\int_0^{\infty} \left(x(t)^T Q x(t) + u(t)^T R u(t)\right) dt$$

- If the control policy is assumed to be time-invariant and linear as described above (i.e., $u(t) = Kx(t)$), minimizing the LQR cost subject to stability constraints can be cast as an SDP and solved using off-the-shelf numerical solvers.

Differentiable Convex Optimization Layers [2]

- We can view deep learning as an instance of differentiable programming
- Compositions of atomic functions
- Each atomic function is differentiable
- We can differentiate through the whole program using the chain rule
- We want to add a convex optimization program as an atom to a deep learning model
- More information:
  - Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., & Kolter, J. Z. (2019). Differentiable convex optimization layers. *Advances in neural information processing systems*, *32*.

ENFORCING ROBUST CONTROL GUARANTEES WITHIN NEURAL NETWORKS

Projecting the Output of a Neural Network to a Safe Set

- Given a dynamical system of the form $\dot{x}(t) \in A(t)x(t) + B(t)u(t) + G(t)w(t)$
- And a quadratic function $V(x) = x^T P x$, let $\mathcal{C}(x)$ denote a set of actions that, for a *fixed* state x, are guaranteed to satisfy the exponential stability condition

$$\mathcal{C}(x) := \{u \in \mathbb{R}^a \mid \dot{V}(x) \leq -\alpha V(x) \quad \forall \dot{x} \in A(t)x + B(t)u + G(t)w\}$$

- We construct a robust nonlinear policy class that *projects* the output of some neural network onto this set

$$\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x)).$$

Optimizing the Neural Network

- We construct a robust nonlinear policy class that *projects* the output of some neural network onto this set

$$\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x)).$$

- Given some performance objective ℓ (e.g., LQR cost)
- Goal: Find parameters θ such that

$$\underset{\theta}{\text{minimize}} \quad \int_0^\infty \ell\,(\,x,\,\pi_\theta(x)\,)\,dt \qquad \text{s.t. } \dot{x} \in A(t)x + B(t)\pi_\theta(x) + G(t)w.$$

Example: Norm-Bounded Linear Differential Inclusions (NLDI)

- Our System: $\dot{x} = Ax(t) + Bu(t) + Gw(t), \|w(t)\|_2 \leq \|Cx(t) + Du(t)\|_2$
- To apply our framework to the NLDI setting, we first compute a quadratic Lyapunov function $V(x) = x^T P x$ by optimizing the LQR cost

$$\int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t)\right) dt$$

- We then use the resultant Lyapunov function to compute the system-specific "safe" set $\mathcal{C}(x)$.

$$\mathcal{C}_{NLDI}(x) := \left\{ u \in \mathbb{R}^a \mid \|Cx + Du\|_2 \leq \frac{-x^T PB}{\|G^T Px\|_2} u - \frac{x^T(2PA + \alpha P)x}{2\|G^T Px\|_2} \right\}$$

- We then create a fast, custom differentiable solver to project onto this set.

Example: Norm-Bounded Linear Differential Inclusions

- The system-specific "safe" set $\mathcal{C}(x)$:

$$\mathcal{C}_{NLDI}(x) := \left\{ u \in \mathbb{R}^a \mid \|Cx + Du\|_2 \leq \frac{-x^T PB}{\|G^T Px\|_2} u - \frac{x^T(2PA + \alpha P)x}{2\|G^T Px\|_2} \right\}$$

- Note that the projection $\mathcal{P}_{\mathcal{C}_{NLDI}(x)}$ represents a projection onto a second-order cone constraint.

$$\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x)).$$

- This projection does not necessarily have a closed form
- We implement it using a differentiable optimization solver

The Second-Order Cone Projection

- The system-specific "safe" set $\mathcal{C}(x)$:

$$\mathcal{C}_{NLDI}(x) := \left\{ u \in \mathbb{R}^a \mid \|Cx + Du\|_2 \leq \frac{-x^T PB}{\|G^T Px\|_2} u - \frac{x^T(2PA + \alpha P)x}{2\|G^T Px\|_2} \right\}$$

$$\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x)).$$

- More Generally, if we consider a set like this:

$$\mathcal{C} = \{x \in \mathbb{R}^n \mid \|Ax + b\|_2 \leq c^T x + d\}$$

- Given an input y, we seek to compute $\mathcal{P}_{\mathcal{C}}(y)$ by solving the problem:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2}\|x - y\|_2^2 \\ \text{subject to} \quad & \|Ax + b\|_2 \leq c^T x + d. \end{aligned}$$

EXPERIMENTS

Experiments: Dynamic Settings

- On five NLDI settings: two synthetic NLDI domains, the cart-pole task, a quadrotor domain, and a microgrid domain.
  - $\dot{x} = Ax(t) + Bu(t) + Gw(t), \|w(t)\|_2 \leq \|Cx(t) + Du(t)\|_2$
  - Generating matrices A, B, G, C and D i.i.d. from normal distributions, and producing the disturbance w(t) using a randomly-initialized neural network
- For each setting, we choose a time discretization based on the speed at which the system evolves, and run each episode for 200 steps over this discretization
- In all cases except the microgrid setting, we use a randomly generated LQR objective

Experiments: Dynamic Settings

- On five NLDI settings: two synthetic NLDI domains, the cart-pole task, a quadrotor domain, and a microgrid domain.
  - In the cart-pole task, the goal is to balance an inverted pendulum resting on top of a cart by exerting horizontal forces on the cart. We linearize this system as an NLDI and add a small additional randomized disturbance satisfying the NLDI bounds
  - Episodes are run for 10 seconds at a discretization of 0.05 seconds.

Experiments: Dynamic Settings

- On five NLDI settings: two synthetic NLDI domains, the cart-pole task, a quadrotor domain, and a microgrid domain.
  - Planar quadrotor. In this setting, our goal is to stabilize a quadcopter in the two-dimensional plane by controlling the amount of force provided by the quadcopter's right and left thrusters. We linearize this system as an NLDI with $D = 0$ and add a small disturbance as in the cart-pole setting.
  - Episodes are run for 4 seconds at a discretization of 0.02 seconds.

Experiments: Dynamic Settings

- On five NLDI settings: two synthetic NLDI domains, the cart-pole task, a quadrotor domain, and a microgrid domain.
  - Microgrid. In this final setting, we aim to stabilize a microgrid by controlling a storage device and a solar inverter.
  - Episodes are run for 4 seconds at a discretization of 0.02 seconds.

Experimental Setup

- $\hat{\pi}_\theta(x) = Kx + \tilde{\pi}_\theta(x)$
- We then optimize our robust policy class $\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x)).$ using two different methods: Robust MBP and Robust PPO

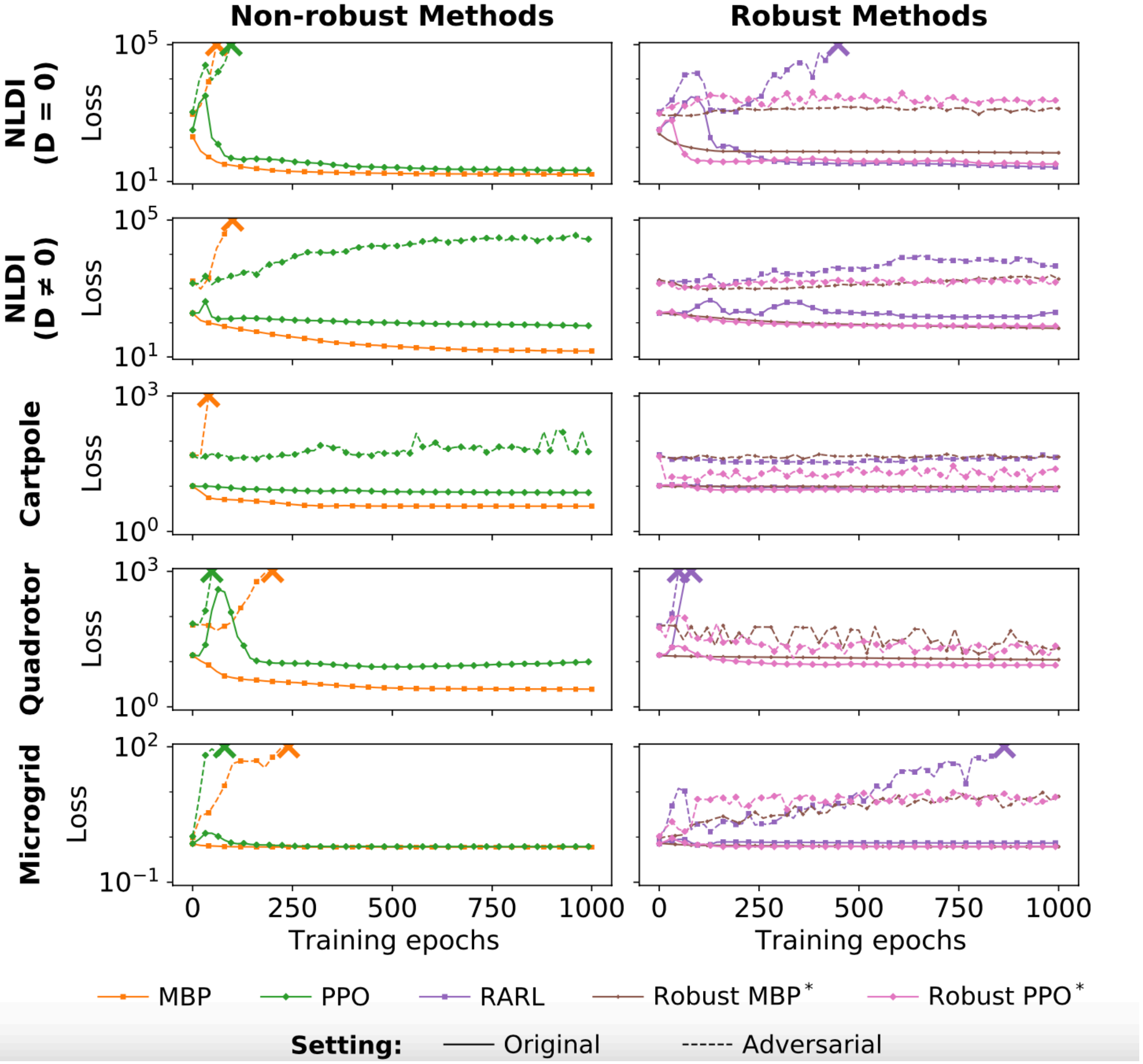
Experimental Setup

- $\hat{\pi}_\theta(x) = Kx + \tilde{\pi}_\theta(x)$
- We then optimize our robust policy class $\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x))$. using two different methods: Robust MBP and Robust PPO
- Baselines:
  - Robust LQR: Robust (linear) controller obtained by minimizing the LQR cost
  - Robust MPC: A robust model-predictive control algorithm based on state-dependent LMIs
  - RARL: The robust adversarial reinforcement learning algorithm
  - LQR: A standard non-robust (linear) LQR controller
  - MBP and PPO
- Two dynamics: Original and Adversarial

Results

Environment		LQR	MBP	PPO	Robust LQR	Robust MPC	RARL	Robust MBP*	Robust PPO*
Generic NLDI	O	373	**16**	21	253	253	**27**	69	33
($D = 0$)	A	—	*unstable*	—	1009	873	*unstable*	1111	2321
Generic NLDI	O	278	**15**	82	199	199	147	**69**	80
($D \neq 0$)	A	—	*unstable*	—	1900	1667	*unstable*	1855	1669
Cart-pole	O	36.3	**3.6**	7.2	10.2	10.2	**8.3**	9.7	8.4
	A	— *unstable*	—	172.1	42.2	47.8	41.2	50.0	16.3
Quadrotor	O	5.4	**2.5**	7.7	13.8	13.8	12.2	11.0	**8.3**
	A	*unstable*	545.7	137.6	64.8	*unstable*†	63.1	25.7	26.5
Microgrid	O	4.59	**0.60**	0.61	0.73	0.73	0.67	**0.61**	**0.61**
	A	—	*unstable*	—	0.99	0.92	2.17	7.68	8.91

Results


Non-robust Methods
Robust Methods
NLDI (D = 0)
NLDI (D ≠ 0)
Cartpole
Quadrotor
Microgrid
Loss
Training epochs
0
250
500
750
1000
MBP
PPO
RARL
Robust MBP*
Robust PPO*
Setting:
Original
Adversarial

References

- Donti, P. L., Roderick, M., Fazlyab, M., & Kolter, J. Z. (2020, September). Enforcing robust control guarantees within neural network policies. In International Conference on Learning Representations.
- Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., & Kolter, J. Z. (2019). Differentiable convex optimization layers. *Advances in neural information processing systems*, *32*.
- The following websites:
  - https://math24.net/method-lyapunov-functions.html
  - https://en.wikipedia.org/wiki/Linear_matrix_inequality