Non-Linear H_∞ Robustness Guarantees for Neural Network Policies

Abstract

Robust control methods ensure system stability under disturbances but often fall short in performance when applied to non-linear systems. Neural-network based control methods trained using deep reinforcement learning (RL) have achieved state-of-the-art performance on many challenging non-linear tasks but often lack robustness guarantees. Prior work proposed a method to enforce robust control guarantees within neural network policies, improving average-case performance over existing robust control methods and worst-case stability over deep RL methods. However, this method assumed linear time-invariant dynamics, which restricts the allowable actions and reduces the flexibility of neural network policies in handling non-linear dynamics. This paper presents a novel approach to enforce non-linear H_∞ robustness guarantees for neural network policies, as well as a tunable robustness parameter that allows for a trading off robustness and average performance, which is an essential feature for real-world deployments. Although the experimental validation of our approach is ongoing, the theoretical foundations presented here aim to facilitate the application of robust control principles to a wider range of non-linear systems, potentially improving both the flexibility and performance of neural network policies in safety-critical applications.

1 Introduction

Robust control methods are essential for ensuring system stability under disturbances but often exhibit limitations in handling non-linear systems effectively. Nonlinear control methods that use deep neural networks, particularly those leveraging deep reinforcement learning (RL), have demonstrated impressive performance on complex, non-linear tasks. However, these methods generally lack robust stability guarantees, which are critical for practical, safety-critical applications. Previous work, notably by Donti et al. (2021), proposed a method that integrates robust control guarantees within neural network policies by projecting the output of a neural network policy onto a set of certifiably stabilizing actions. This set of actions was constructed using robust control techniques for linear time-invariant dynamics with potentially non-linear disturbances. This method achieved a notable balance between improving average-case performance over traditional robust controllers and enhancing worst-case stability over non-robust neural-network policies. However, the assumption of linear time-invariant dynamics considerably restricts the allowable actions and reduces the flexibility of neural network policies in handling non-linear systems. Additionally, the reliance on a predefined dynamics model is less compatible with the data-driven nature of RL. In this paper, we address these limitations by presenting a novel approach that generalizes that of Donti et al. (2021) to achieve non-linear H_∞ robustness guarantees for neural network policies. Our contributions are threefold:

1. **Nonlinear Robustness Projection**: We develop a projection method that accommodates non-linear dynamics and controllers, thereby expanding the set of allowable actions without compromising robustness. This generalization allows our approach to handle a broader class of systems beyond the linear approximations used in previous methods.

Preprint. Under review.

2. **Data-Driven Robustness Framework**: We introduce a robustness framework that learns the dynamics model from data, which aligns more closely with the principles of RL. This eliminates the need for a predefined system model, making our approach more adaptable to various practical scenarios.

3. **Tunable Robustness/performance trade-off**: Our methodology includes a tunable parameter that trades-off robustness and average performance, by determining the rate of *energy dissipation* of the controlled system. Such a parameter can serve as a control knob, allowing practitioners to balance robustness and average performance according to specific real-world application requirements, where trade-offs between performance and stability must be carefully managed.

Although the experimental validation of our approach is ongoing, the theoretical foundations presented here aim to facilitate the application of robust control principles to a wider range of non-linear systems, potentially improving both the flexibility and performance of neural network policies in safety-critical applications.

2 Background

Control theory deals with the design and analysis of controllers for dynamical systems. A dynamical system is characterized by its state, typically a vector of real values that describe the system at a given moment in time. The goal of a controller is to map each system state x to a control action u to optimize some performance metric over the system's lifetime, often represented as a cost that accumulates over time.

Linear control focuses on developing linear controllers for systems with linear dynamics. In control theory, particularly in robust control contexts, the dynamics of a linear system are often described by the following representation:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + Gw(t), \quad w \in \mathcal{L}_2 \\ z(t) &= Cx(t) + Du(t) \end{aligned} \tag{1}$$

where t denotes time, $x(t) \in \mathbb{R}^n$ denotes the state vector, $u(t) \in \mathbb{R}^m$ represents the control input, and $w(t) \in \mathbb{R}^p$ encapsulates external disturbances or modeling uncertainties. The matrices A, B, and G are of appropriate dimensions, and w is assumed to belong to the space of signals with finite $\mathcal{L}_2$ norm, ensuring bounded energy. The output z is typically related to the performance or the cost accumulated at each step, where C and D are matrices of appropriate dimensions.

Non-linear control focuses on developing non-linear controllers for systems with non-linear dynamics. One way to define a non-linear dynamic system is the following:

$$\begin{aligned} \dot{x}(t) &= f(x(t)) + g_2(x(t))u(t) + g_1(x(t))w(t) \\ z(t) &= h_1(x(t)) + k_{12}(x(t))u(t), \quad w \in \mathcal{L}_2 \end{aligned} \tag{2}$$

where $x \in X$ is a state vector, t, u, w are as in Equation 1, and the functions $f : X \to C^\infty(X)$, $g_1 : X \to M_{n\times p}(X)$, $g_2 : X \to M_{n\times m}(X)$, $h_1 : X \to \mathbb{R}^s$, and $k_{12} : X \to M_{n\times m}(X)$ are assumed to be real C^∞-functions of x.

A special case of non-linear systems is given by the following polynomial dynamics representation (omitting dependency on t for brevity):

$$\begin{aligned} \dot{x} &= A(x)x^{\{d\}} + B_2(x)u + B_1(x)w \\ z &= C_1(x)x^{\{d\}} + D_{12}(x)u \end{aligned} \tag{3}$$

where $x^{\{d\}}$ is a vector of monomials of degree $\leq d$ in x, and $A(x)$, $B_1(x)$, $B_2(x)$, $C_1(x)$, and $D_{12}(x)$ are matrices whose entries are (potentially constant) functions of x. Note that while $A(x)$ in Equation 1 is an $n \times n$ matrix, $A(x)$ in Equation 3 is a $\{d\} \times \{d\}$ matrix where $\{d\}$ is the number of monomials in $x^{\{d\}}$,

2.1 Robust H_∞ Control for Non-Linear Systems

Robust control addresses uncertainties arising from external disturbances and incomplete knowledge of system parameters. The goal is to design a controller that maintains performance despite these uncertainties. This paper focuses on uncertainties arising from external disturbances. The H_∞ control problem is a robust control approach that aims to minimize the worst-case effect of disturbances on the system. This is formalized using the H_∞ norm, which quantifies the maximum gain from the disturbance input w to the controlled output z. The H_∞ control problem can be stated as follows: find a controller $K(x)$ such that for some $\gamma > 0$ the controlled system satisfies

$$\int_0^\infty \|z(t)\|^2 dt \leq \gamma^2 \left(\|x(0)\|^2 + \int_0^\infty \|w(t)\|^2 dt \right), \quad \forall w \in \mathcal{L}_2 \tag{4}$$

where $\|x(0)\|$ is the initial energy state. A system satisfying this condition is said to have an $\mathcal{L}_2$-*gain* from w to z that is less than or equal to γ.

In control theory, particularly in the context of H_∞ control, the concept of *dissipation* is used to analyze the stability and performance of a system. The *dissipation inequality* is given by:

$$S(x(t_1)) \leq S(x(t_0)) + \int_{t_0}^{t_1} s(w(t), z(t))\, dt \tag{5}$$

where $S : X \to \mathbb{R}_+$ is *storage function* representing the system's stored energy, t_0 and t_1 are the initial and final times, respectively, and $s(w(t), z(t))$ is a *supply rate* quantifying the energy exchange with the environment. A system represented by Equation 1 or 2, is said to be *dissipative* with respect to a supply rate $s(w(t), z(t))$ if it satisfies Equation 5 for all $t_1 \geq t_0$, $x(t_0), x(t_1) \in X$, and all disturbances $w \in \mathcal{L}_2$. Equation 5 provides a way to ensure that the system dissipates more energy than it absorbs from external sources, which include the contributions of the system state, control input, and external disturbances.

Choosing the supply rate as:

$$s(w, z) = \frac{\gamma^2}{2}\|w\|^2 - \frac{1}{2}\|z\|^2, \quad \gamma \geq 0 \tag{6}$$

a system is dissipative with respect to this supply rate if there exists a storage function $S(x)$ such that:

$$S(x(t_1)) \leq S(x(t_0)) + \int_{t_0}^{t_1} \left(\frac{\gamma^2}{2}\|w(t)\|^2 - \frac{1}{2}\|z(t)\|^2 \right) dt \tag{7}$$

and if so, it can be seen that it has an $\mathcal{L}_2$-gain of γ. We therefore would like to develop systems that are dissipative with respect to this supply rate. To ensure the system is robust to external influences and remains stable, the dissipation inequality must hold for all times, states, and disturbances. By subtracting $S(x(t_0))$ from both sides of Equation 7 and letting $t_1 \to t_0$, we get:

$$\frac{dS}{dt} = S_x(x)\dot{x}(t) \leq \frac{\gamma^2}{2}\|w(t)\|^2 - \frac{1}{2}\|z(t)\|^2 \tag{8}$$

where $S_x(x)$ denotes the gradient of S with respect to x.

Next, using $S(x)$, a non-negative function $V(x) = S(x) - S(x^*)$ is defined around a strict local minimum x^* of $S(x)$. Defining $V(x)$ is beneficial because it allows us to use $V(x)$ as a *Lyapunov function*, which provides a measure of the system's stability. A Lyapunov function is a non-negative scalar function that decreases over time for a stable system, indicating that the system's state is approaching an equilibrium point. For the dynamical systems specified in Equations 1 and 3, and quadratic positive-definite Lyapunov functions $V(x) = x^\top Px$ and $V(x) = x^{\{d\}T}Px^{\{d\}}$ respectively, it is possible to use semidefinite programming to construct linear controllers $u(t) = Kx(t), K \in M_{m\times n}$ and $u(t) = K(x)x^{\{d\}}(t), K(x) \in M_{m\times\{d\}}$, ensuring dissipativity and $\mathcal{L}_2$-gain, as we describe in Section 3.

By (i) using the fact that $V_x(t) = (S(x) - S(x^*))_x(t) = S_x(t)$, (ii) substituting into Equation 8 the general non-linear system dynamics of Equation 2 $\dot{x} = f(x) + g_2(x)u + g_1(x)w$, (iii) omitting t for brevity, and (iv) rearranging terms, we get:

$$V_x(f(x) + g_2(x)u + g_1(x)w) - \frac{\gamma^2}{2}\|w\|^2 + \frac{1}{2}\|z(x,u)\|^2 \leq 0 \tag{9}$$

To ensure that the system is robust to disturbances and optimally controlled, the left-hand side is maximized with respect to w to account for the worst-case disturbance, and minimized with respect to u to find the optimal control input. Maximizing with respect to w results in $w^* = \frac{1}{\gamma^2} g_1(x)^\top V_x^\top$, and minimizing with respect to u results in $u^* = -g_2(x)^\top V_x^\top$. Substituting these into the above inequality and assuming that $h_1(x)^\top k_{12}(x) = 0$ yields the Hamilton-Jacobi inequality (HJI):

$$\text{HJI:} \quad V_x f(x) + \frac{1}{2} V_x \left(\frac{1}{\gamma^2} g_1(x) g_1(x)^\top - g_2(x) k_{12}(x)^\top k_{12}(x) g_2(x)^\top \right) V_x^\top + \frac{1}{2} h_1(x)^\top h_1(x) \leq 0 \tag{10}$$

This inequality must be satisfied for all $x \in X$. Solving it amounts to finding a Lyapunov function V, and therefore a robust controller u^* for which the inequality holds for all system states. Such a controller renders a dissipative controlled system, which therefore has an $\mathcal{L}_2$ gain $\leq \gamma$. In this general form, solving the HJI inequality is NP-hard. However, for systems in Equation 3 (and 1 as a special case of 3), the HJI inequality can be reduced to a semidefinite program, for which there are efficient solution methods, as described in Section 3.

2.2 Robust Control Guarantees for Neural Network Policies

Reinforcement Learning (RL) Sutton and Barto (2018) is an approach for learning a mapping from states to actions, known as *policies*, without requiring a model of the system dynamics. Unlike traditional control methods, which rely on explicit models, RL focuses on optimizing actions through trial and error to maximize cumulative rewards, which is typically equivalent to minimizing cumulative cost.

While robust control methods offer rigorous guarantees on system stability under disturbances, they often focus on linear dynamics and controllers, and therefore result in simple controllers with poor average-case performance. In contrast, nonlinear control methods trained using *deep RL* (RL that uses deep neural networks) have achieved state-of-the-art average-case performance on a variety of challenging domains, however they lack robustness guarantees.

Donti et al. (2021) proposed to apply techniques from robust linear control to enforce robustness on deep RL policies. Their paper addresses the challenging tradeoff between robustness and performance in safety-critical systems. Their technique combines the strengths of both approaches by constructing a generic nonlinear control policy class parameterized by neural networks, that enforces the same provable robustness criteria as robust control. The resulting policies improved average-case performance over existing robust control methods and worst-case stability over non-robust deep RL methods.

More specifically, the approach of Donti et al. (2021) leverages the framework of linear robust control, focusing on systems that are variations of the system described by Equation 1, specifically:

- Norm-bounded Linear Differential Inclusions (NLDIs):
$$\dot{x} = Ax(t) + Bu(t) + Gw(t), \quad \|w(t)\|_2 \leq \|Cx(t) + Du(t)\|_2$$
- Polytopic Linear Differential Inclusions (PLDIs):
$$\dot{x}(t) = A(t)x(t) + B(t)u(t), \quad (A(t), B(t)) \in \text{Conv}\{(A_1, B_1), \ldots, (A_L, B_L)\}.$$
- H_∞ control setting with linear time-invariant dynamics, given by Equation 1:
$$\dot{x}(t) = Ax(t) + Bu(t) + Gw(t), \quad w \in \mathcal{L}_2$$

All of these systems are linear in their state and control variables, with potentially non-linear disturbances, meaning that the matrices A and B are constant or time-varying but linear, while the disturbance w and the uncertainty in PLDIs can be arbitrary but are bounded Boyd et al. (1994). The key steps of their approach are as follows:

1. **Define the Performance Objective:** The performance objective is specified, typically as minimizing a cost function over time.
2. **Define the Stability Requirement:** The stability requirement ensures that the system remains stable under disturbances, often represented by a Lyapunov function.
3. **Define a Policy Optimizer:** A policy optimizer is used to adjust the neural network parameters to meet the defined objectives.
4. **Solve the Linear Matrix Inequality (LMI):** For systems of the three types mentioned above (namely NLDI, PLDI, and those given by Equation 1), the Hamilton-Jacobi Inequality (HJI) reduces to a Linear Matrix Inequality (LMI), the solution of which are constant matrices P and K that satisfy the robustness constraints for systems with linear dynamics.
5. **Construct a Set of Allowable Robust Actions $\mathcal{C}(x)$ Using P:** The solution P, K from the LMI is used to construct the set $\mathcal{C}(x)$ of actions satisfying the robustness constraint.
6. **Construct a Robust Policy Class using $\mathcal{C}(x)$:** The set $\mathcal{C}(x)$ is used to construct a robust nonlinear policy class that projects the output of some neural network onto this set. Formally, given an arbitrary nonlinear (neural network-based) policy class $\hat{\pi}_\theta : \mathbb{R}^s \to \mathbb{R}^a$ parameterized by θ, and a projection operator $\mathcal{P}(\cdot)$ for some set $(\cdot)$, the robust policy class is defined as as $\pi_\theta : \mathbb{R}^n \to \mathbb{R}^m$, where
$$\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x)).$$
7. **Train the Neural Network Policy:** The policy optimizer trains the neural network policy $\pi_\theta(x)$ to optimize the performance objective. The projection operator $\mathcal{P}_{\mathcal{C}(x)}$ which is part of the policy $\pi_\theta(x)$ ensures that the policy satisfies the robust control constraints.

This approach is summarized in Algorithm 1:

Algorithm 1 Learning Provably Robust Controllers with Deep RL Donti et al. (2021)

Input: Performance objective l, stability requirement, policy optimizer $\mathcal{A}$
Solve the Linear Matrix Inequality (LMI) for matrices P, K
Construct a set of allowable robust actions $\mathcal{C}(x)$ using P, K
Construct a robust policy class by projecting $\hat{\pi}_\theta(x)$ onto $\mathcal{C}(x)$: $\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x))$
Train π_θ via $\mathcal{A}$ to optimize l
Return: Optimized policy π_θ

3 Framework for Non-Linear H_∞ Robustness in Neural Network Policies

Despite the significant advancements provided by Donti et al. (2021), their approach has a few notable limitations. By leveraging linear time-invariant dynamics, the method restricts the class of systems to those that can be approximated linearly, which may not be realistic for many practical applications involving inherently nonlinear dynamics. This assumption of linearity leads to a constrained set of allowable actions, potentially limiting the performance of the control system in setups with non-linear dynamics. Additionally, the requirement for a predefined dynamics model makes this approach less adaptable to reinforcement learning (RL), which inherently thrives on learning from data without explicit system models. This section addresses these shortcomings by proposing a methodology that eliminates the need for a priori system models, accommodates nonlinear dynamics, and introduces tunable robustness parameters, thereby expanding the potential for robust, high-performance neural network policies deployed in the real-world.

3.1 System Dynamics Specification

To model nonlinear dynamics, we use the polynomial dynamics defined by Equation 3:

$$\dot{x} = A(x)x^{\{d\}} + B_2(x)u + B_1(x)w,$$
$$z = C_1(x)x^{\{d\}} + D_{12}(x)u,$$

where $x^{\{d\}}$ is a vector of monomials of degree $\leq d$ in x. This form has the advantages of (i) being highly expressive, capable of representing a wide range of complex dynamics, (ii) being learnable from data using, e.g., self-supervised learning, and (iii) making the solution of the Hamilton-Jacobi inequality (HJI) tractable, as described in Section 3.2.

If the polynomial dynamics are known, we directly use the matrices $A(x)$, $B_2(x)$, and $B_1(x)$ to define the system. If the dynamics are unknown, we follow these steps:
1. **Data Collection**: Gather data from system execution via simulations or real-world experiments.
2. **Model Fitting**: Fit a polynomial dynamics model from the collected data using an appropriate machine learning approach, e.g., self-supervised learning, obtaining matrices $A(x)$, $B_1(x)$, and $B_2(x)$. Note that $C_1(x)$ and $D_{12}(x)$ define the system's cost function which is typically known in control applications, but could similarly be learned if needed.

Once we have a given or learned system model in the form of Equation 3, we proceed to execute an algorithm similar to Algorithm 1, with a few key modifications to accommodate nonlinear dynamics and controllers. The main modification uses a method to make the solution of Hamilton-Jacobi inequality (HJI) tractable, described next.

3.2 Hamilton-Jacobi Inequality (HJI) for Systems with Polynomial Dynamics

In Section 2.1, we have seen that solving the HJI (Equation 10) results in a Lyapunov function V that defines a robust controller u^* with an $\mathcal{L}_2$ gain of γ. While solving the general form of the HJI is NP-hard, there is an efficient method for solving it for systems with polynomial dynamics. Theorem 1, as presented here, is a well-established result that is fundamental to our approach. Its proof can be found, for example, in the work of Capua et al. (2013).

Theorem 1. *Let $\mathcal{F}$ be a system with polynomial dynamics specified in Equation 3, $x^{\{d\}}$ be a $\{d\} \times 1$ vector of all monomials of degree $\leq d$ in x, and $M(x)$ be a $\{d\} \times n$ polynomial matrix whose (i,j)-th entry is given by*

$$M_{ij} = \frac{\partial x_i^{\{d\}}}{\partial x_j}, \quad i = 1, \ldots, \{d\}, \quad j = 1, \ldots, n \tag{11}$$

Assume the system's cost components are orthogonal, i.e. $C_1(x)^\top D_{12}(x) = 0$, and that $R_2(x) = D_{12}(x)^\top D_{12}(x) > 0$. Then the following two claims hold:

1. *For the system $\mathcal{F}$, the HJI inequality becomes the following State-Dependent Linear Matrix Inequality (SDLMI):*

$$\begin{bmatrix} Y(x)^\top B_2(x)^\top M(x)^\top + M(x)B_2(x)Y(x) + SA(x)^\top M(x)^\top + M(x)A(x)S & M(x)B_1(x) & Y(x)^\top & SC_1(x)^\top \\ * & -\gamma^2 I & 0 & 0 \\ * & * & -R_2(x) & 0 \\ * & * & * & -I \end{bmatrix} \preceq 0 \tag{12}$$

   where $*$ indicates symmetric entries in a symmetric matrix and $\preceq$ indicates negative semi-definiteness. Solving it amounts to finding matrices $S = S^\top > 0$ and $Y(x)$ such that the matrix on the left hand side is negative semi-definite.

2. *If there exist matrices $S = S^\top > 0$ and $Y(x)$ that solve this version of HJI while minimizing γ, then for $P = S^{-1}$ and $K(x) = Y(x)P$, the controller $u = K(x)x^{\{d\}}$ renders a dissipative, stable system with an $\mathcal{L}_2$-gain $\leq \gamma$. In that case, the Lyapunov function is $V(x) = x^{\{d\}\top} P x^{\{d\}}$.*

This formulation provides a robust framework for analyzing and designing controllers for non-linear systems that ensure system stability and performance, making it a powerful tool in control theory. For linear systems as described by Equation 1, the matrix $M(x)$ becomes the identity matrix, and Equation 12 simplifies to a Linear Matrix Inequality (LMI), which is a convex semidefinite program for which there are many open-source and commercial solvers available.

For systems described by Equation 3, the entries of $M(x)$ are polynomial functions. Equation 12 is a non-convex SDLMI, and solving it involves polynomial non-negativity conditions, which are NP-hard to test and construct. However, if we replace the non-negativity conditions with *Sum-of-Squares* (SOS) conditions – meaning that a polynomial can be expressed as a sum of squared polynomials – both testing and constructing a Lyapunov function $V(x) = x^{\{d\}T} P x^{\{d\}}$, and therefore a robust controller $u = K(x)x^{\{d\}}$, can be done efficiently using a type of semidefinite programming known

as Sum-of-Squares Programming (SOSP) Papachristodoulou and Prajna (2005); Capua et al. (2013), for which there are available solvers such as SOSTools Papachristodoulou et al. (2013) and its Python version, SOSPy.

3.3 Constructing a Set of Actions Satisfying Non-Linear H_∞ Robustness

Next, we use the solution of SDLMI 12, namely P and $K(x) = Y(x)P$, to construct the set of actions $\mathcal{C}_{\text{NL-}H_\infty}(x)$ satisfying non-linear H_∞ robustness to which the outputs of the neural network are projected. This projection ensures that the neural network policy adheres to the robustness criteria dictated by the nonlinear H_∞ framework. Our approach generalizes the methodology proposed by Donti et al. (2021) in Algorithm 1 from linear time-invariant dynamics to systems with polynomial dynamics by adapting it to accommodate the state-dependent linear matrix inequality (SDLMI) described in Equation 12.

The robust action set $\mathcal{C}_{\text{NL-}H_\infty}(x)$ is defined as the set of control actions u that satisfy Equation 9 with the worst disturbance w^*, which was also used in the HJI Equation 10. Substituting the polynomial dynamics from Equation 3 along with $w^* = \frac{1}{\gamma^2} g_1(x)^\top V_x^\top = \frac{1}{\gamma^2} B_1(x)^\top V_x^\top$ into Equation 9, we get:

$$V_x \left(A(x)x^{\{d\}} + B_2(x)u + B_1(x)\frac{1}{\gamma^2} B_1(x)^\top V_x^\top \right) - \frac{1}{2\gamma^2}\|B_1(x)^\top V_x^\top\|^2 + \frac{1}{2}\|C_1(x)x^{\{d\}} + D_{12}(x)u\|^2 \leq 0 \tag{13}$$

The gradient with respect to x of the Lyapunov function $V(x) = x^{\{d\}\top} P x^{\{d\}}$ resulting from solving the SDLMI is $V_x = 2x^{\{d\}\top} PM(x)$. Substituting this into the above equation and using the common assumptions $C_1(x)^\top D_{12}(x) = 0$, $Q(x) = C_1(x)^\top C_1(x) \succ 0$, and $R_2(x) = D_{12}(x)^\top D_{12}(x) \succ 0$, we get:

$$\begin{aligned} 2x^{\{d\}\top} PM(x) \Bigg(A(x)x^{\{d\}} + B_2(x)u + \frac{1}{\gamma^2} B_1(x)B_1(x)^\top 2M(x)^\top P x^{\{d\}} \Bigg) \\ - \frac{2}{\gamma^2} x^{\{d\}\top} PM(x)B_1(x)B_1(x)^\top M(x)^\top P x^{\{d\}} \\ + \frac{1}{2} x^{\{d\}\top} Q x^{\{d\}} + \frac{1}{2} u^\top R_2(x) u \leq 0 \end{aligned}$$

Simplifying and rearranging terms, we get:

$$\begin{aligned} x^{\{d\}\top} \left(2PM(x)A(x) + \frac{2}{\gamma^2} PM(x)B_1(x)B_1(x)^\top M(x)^\top P + \frac{1}{2} Q \right) x^{\{d\}} \\ + 2x^{\{d\}\top} PM(x)B_2(x)u + \frac{1}{2} u^\top R_2(x) u \leq 0 \end{aligned}$$

This can be rewritten as:

$$\begin{aligned} x^{\{d\}\top} \left(PM(x)A(x) + A(x)^\top M(x)^\top P + \frac{2}{\gamma^2} PM(x)B_1(x)B_1(x)^\top M(x)^\top P + \frac{1}{2} Q \right) x^{\{d\}} \\ + \left(2B_2(x)^\top M(x)^\top P x^{\{d\}} \right)^\top u + \frac{1}{2} u^\top R_2(x) u \leq 0 \end{aligned}$$

Denoting:

$$\tilde{P} := \frac{1}{2}R_2(x),$$
$$\tilde{q} := 2B_2(x)^\top M(x)^\top P x^{\{d\}},$$
$$\tilde{r} := x^{\{d\}\top}\left(PM(x)A(x) + A(x)^\top M(x)^\top P + \frac{2}{\gamma^2}PM(x)B_1(x)B_1(x)^\top M(x)^\top P + \frac{1}{2}Q\right)x^{\{d\}},$$

we define the set of actions satisfying non-linear H_∞ robustness as:

$$\mathcal{C}_{\text{NL-}H_\infty}(x) = \{u \in \mathbb{R}^m \mid u^\top \tilde{P} u + 2\tilde{q}^\top u + \tilde{r} \leq 0\} \tag{14}$$

which generalizes equations B.3 and B.4 from Donti et al. (2021) from the case of linear time-invariant dynamics to the case of polynomial dynamics. The set $\mathcal{C}_{\text{NL-}H_\infty}(x)$ is non-empty, as the controller $u = K(x)x^{\{d\}} = Y(x)Px^{\{d\}}$ derived from the solution $Y(x), P$ of the SDLMI 12 satisfies Equation 13 by construction. Additionally, the set $\mathcal{C}_{\text{NL-}H_\infty}(x)$ is an ellipsoid in u , and therefore the projection $\mathcal{P}_{\mathcal{C}_{\text{NL-}H_\infty}(x)}$ can be viewed as a second-order cone projection and implemented using the fast custom differentiable optimization solver described in Appendices B.2 and C in Donti et al. (2021). Given an arbitrary neural network-based policy class $\hat{\pi}_\theta$ and the above implementation of the projection operator $\mathcal{P}_{\mathcal{C}_{\text{NL-}H_\infty}(x)}$ a robust policy class is constructed, which adheres to the robustness criteria dictated by the nonlinear H_∞ framework:

$$\pi_\theta(x) = P_{\mathcal{C}_{\text{NL-H}\infty}(x)}(\hat{\pi}_\theta(x)).$$

Our overall approach is summarized in Algorithm 2.

Algorithm 2 Learning Neural Network Controllers with Non-Linear H_∞ Robustness Guarantees

1: **Fixed:** Set the robustness requirement to be Non-linear H_∞
2: **Input:** Performance objective l, tunable robustness parameter γ, policy optimizer A
3: **Specify or learn** polynomial system dynamics ▷ Section 3.1
4: **Solve** the SDLMI (12) using γ, for $P, Y(x), K(x)$ satisfying SOS constraints ▷ Section 3.2
5: **Construct** a set of allowable robust actions $\mathcal{C}_{\text{NL-}H_\infty}(x)$ using $P, Y(x)$ and $K(x)$ ▷ Section 3.3
6: **Construct** a robust policy class by projecting $\hat{\pi}_\theta(x)$ onto $\mathcal{C}_{\text{NL-}H_\infty}(x)$: $\pi_\theta(x) = \mathcal{P}_{\mathcal{C}_{\text{NL-}H_\infty}}(\hat{\pi}_\theta(x))$
7: **Train** π_θ via A to optimize l
8: **Return:** Optimized policy π_θ

3.4 Tuning Gamma for Trade-off Between Robustness and Performance

The parameter γ in Algorithm 2 offers a tunable trade-off between robustness and average performance. By adjusting γ, which controls the system's energy dissipation, or $\mathcal{L}_2$-gain, one can effectively balance the robustness of the control policy against disturbances and the average performance of the neural network policy. Lower values of γ increase robustness by limiting the $\mathcal{L}_2$-gain, thereby reducing the system's sensitivity to disturbances, but may limit the overall performance. Conversely, higher values of γ can enhance performance but reduce robustness. This tunability is critical for practical applications where both robustness and performance are important. Hence, γ serves as a knob to fine-tune the desired balance between robustness and performance in the control system.

4 Conclusions

In this paper, we have introduced a novel approach for ensuring non-linear H_∞ robustness in neural network policies, addressing key limitations of previous methods, particularly the reliance on linear time-invariant dynamics and predefined system models. Our contributions include a non-linear robustness projection method, a data-driven robustness framework, and a tunable robustness parameter that trades-off robustness and average performance by controlling the energy dissipation of the controlled system. These advancements facilitate the application of robust control principles to a wider range of non-linear systems, improving both the flexibility and performance of neural

network policies in safety-critical applications. Although the experimental validation is ongoing, the theoretical foundations laid out in this work suggest potential applicability for future real-world applications.

Future work will focus on an empirical analysis that will assess the performance of our framework and its potential real-world applicability, specifically in complex high-dimensional systems. Integrating this approach with a variety of advanced RL techniques seems like a promising avenue for developing robust and adaptive control strategies. By continuing to expand and improve upon the foundation laid in this paper, we aim to provide robust, high-performance control framework that supports the demands of modern autonomous systems.

References

Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. 1994. *Linear matrix inequalities in system and control theory*. SIAM.

Alon Capua, Nadav Berman, Amir Shapiro, and Daniel Choukroun. 2013. Nonlinear Output-Feedback H_∞ Control for Spacecraft Attitude Control. In *Advances in Aerospace Guidance, Navigation and Control: Selected Papers of the Second CEAS Specialist Conference on Guidance, Navigation and Control*. Springer, 139–158.

Priya L. Donti, Melrose Roderick, Mahyar Fazlyab, and J. Zico Kolter. 2021. Enforcing robust control guarantees within neural network policies. arXiv:2011.08105 [cs.LG]

A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. A. Parrilo. 2013. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*. `http://arxiv.org/abs/1310.4716`. Available from `http://www.eng.ox.ac.uk/control/sostools`, `http://www.cds.caltech.edu/sostools` and `http://www.mit.edu/~parrilo/sostools`.

Antonis Papachristodoulou and Stephen Prajna. 2005. A tutorial on sum of squares techniques for systems analysis. In *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2686–2700.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.